TREBALL DE FI DE GRAU

TFG TITLE: CFD study of the bubble generation process in a T-junction with inversed flows

DEGREE: Grau en Enginyeria d'Aeronavegació

AUTHOR: Blanca Dalfó Ferrer

ADVISORS: Santiago Arias Calderón
Ming-Jyh Chern

DATE: July 19, 2018

Title : CFD study of the bubble generation process in a T-junction with inversed flows

Author: Blanca Dalfó Ferrer

Advisors: Santiago Arias Calderón
Ming-Jyh Chern

Date: July 19, 2018

Overview

Two-phase flow applications are of great interest in the space sector thanks to its advantages, in the recent years, the research on this subject has increased considerably. Some examples can be found in space bioreactors, chemical contactors, life-support systems for human exploration and development of space, etc. Despite the great effort on understanding the behaviour of two-phase flows in microgravity, there are still many unknowns, therefore, further research is needed.

In this study, we have dedicated our work to analyse the bubble generation of the water-air combination in a T-junction capillary. The simulations have been performed in a microgravity environment due to the very different behavior when compared to the one observed in the presence of gravitational forces, as well as, despite there is a lot of research in this field, there is still much more to be done.
The open source program OpenFoam is used as the main tool for our simulations and Paraview as a post-processing tool. The InterFoam solver is selected to use in order to simulate laminar flow with two incompressible and isothermal phases. The study of numerical simulations has been carried out to compare it with experimental data and therefore, validate the software for this use.

Before obtaining the results, convergence tests on the mesh have been performed, in addition to a study to detach the bubbles from the walls, focusing on the contact angle and wettability condition. We validated our mesh and selected the best boundary conditions to perform the finals tests. In the results, we carried out the simulations for three groups of combinations of velocities with $U_{SL} = 0.106$ m/s, $U_{SL} = 0.318$ m/s, and $U_{SL} = 0.531$ m/s and different gas superficial velocity to analyse the behaviour of the bubble frequency, volume, velocity, and longitude. We obtained that simulations and experiments are really similarly qualitatively, with some differences quantitatively. Besides, they adequately reproduced the bubbles formation.
By analysing all the parameters, we observe that for lower liquid superficial velocities the simulations approach perfectly to the experiments while increasing its velocity the simulations start having more fluctuations and move away from the experimental data.

Títol: Estudi CFD del procés de generació de bombolles en un capil·lar en forma de "T" amb els fluxes invertits

Autor: Blanca Dalfó Ferrer

Directors: Santiago Arias Calderón
Ming-Jyh Chern

Data: 19 de juliol de 2018

Resum

Les aplicacions de flux bifàsic són de gran interès en el sector espacial gràcies als seus avantatges, en els últims anys, la recerca sobre aquest tema ha augmentat considerablement. Alguns exemples es poden trobar en bioreactors espacials, contactors químics, sistemes de suport vital per a l'exploració humana i el desenvolupament de l'espai, etc. Malgrat el gran esforç per comprendre el comportament dels fluxos bifàsics en un ambient de microgravetat, encara hi ha moltes incògnites, per això, és necessària més investigació.

En aquest estudi, hem dedicat el nostre treball a analitzar la generació de bombolles de la combinació d'aigua i gas en un capil·lar en forma de "T". Les simulacions s'han realitzat en un entorn de micro gravetat per la gran diferència de comportament en comparació amb l'observat en presència de forces gravitacionals, així com, tot i que hi ha molta investigació en aquest camp, encara queda molta més per fer.
El programa de codi obert OpenFoam s'utilitza com a eina principal per a les nostres simulacions i Paraview com a eina de postprocessament. S'ha seleccionat el *solver* InterFoam, dissenyat per simular fluxos multifàsics, laminars, incompressibles i isotèrmics. L'estudi de simulacions numèriques s'ha dut a terme per comparar-ho amb dades experimentals i d'aquesta manera poder validar el software per aquest ús.

Abans d'obtenir els resultats, s'han realitzat proves de convergència sobre el nombre de cel·les de la malla, així com un estudi per separar les bombolles de les parets, centrant-se en l'angle de contacte i la condició de *wettability*. Hem validat la malla i seleccionat les millors condicions de contorn per realitzar els tests finals. En els resultats, hem dut a terme les simulacions per tres grups de combinacions de velocitat amb $U_{SL} = 0.106$ m/s, $U_{SL} = 0.318$ m/s i $U_{SL} = 0.531$ m/s i diverses velocitats superficials del gas per estudiar el comportament de la freqüència, volum, velocitat i longitud de la bombolla. Hem obtingut que les simulacions i els experiments són molt similars qualitativament, amb algunes diferències quantitativament. A més, reprodueixen adequadament la formació de les bombolles.
Analitzant tots els paràmetres, observem que per a una velocitat superficial del líquid baixa, les simulacions s'aproximen molt als experiments mentre que per velocitats més altes apareixen més fluctuacions i els valors s'allunyen dels experimentals.

To my family, friends, and
of course, to my professors

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Dimensionless number

B_0 Bond number

C_a Capillary number

R_e Reynolds number

W_e Weber number

Greek Symbols

α Mean void fraction

Δ_t Time step

Δ_x Cell size

μ_G Gas dynamic viscosity

μ_G Mean viscosity

μ_L Liquid dynamic viscosity

$\overline{\sigma_f}$ Bubble frequency normalised standard deviation

$\overline{\sigma_{L_B}}$ Bubble length normalised standard deviation

$\overline{\sigma_{U_B}}$ Bubble velocity normalised standard deviation

$\overline{\sigma_{V_B}}$ Bubble volume normalised standard deviation

ϕ_c Diameter of a circular capillary channel

ρ Mean density

ρ_G Gas density

ρ_L Liquid density

σ Surface tension

σ_f Bubble frequency standard deviation

σ_{L_B} Bubble length standard deviation

σ_{U_B} Bubble velocity standard deviation

σ_{V_B} Bubble volume standard deviation

Roman Symbols

$\overline{L_B}$ Normalized bubble length

$\overline{U}_B$ Normalized bubble velocity

A Cross-sectional area

a_0 Initial slope

C_0 Void fraction distribution coefficient

C_1, C_2 Fitting constants for the bubble length

d	Distance between the two surfaces (1mm)
E_f	Bubble frequency error
E_{L_B}	Bubble length error
E_{U_B}	Bubble velocity error
E_{V_B}	Bubble volume error
f	Bubble generation frequency
F_s	Surface tension force
f_{sat}	Saturation frequency
g	Gravitational acceleration
k	Mean curvature
L_B	Bubble length
p	Pressure
Q_G	Gas flow rate
T_e	Time the bubble enters the monitor
T_f	Time to cross from the first monitor to the second
T_l	Time the bubble leaves the monitor
T_s	Bubble period
U	Mean velocity
U_B	Bubble velocity
U_{SG}	Superficial gas velocity
U_{SL}	Superficial liquid velocity
V_B	Bubble volume
x, y, z	Cartesian coordinates

ACKNOWLEDGEMENTS

First and foremost, I have to thank my advisors, Dr. Ming-Jyh Chern and Dr. Santiago Arias Calderón. Without their assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished. My professor from Spain, Santiago Arias offered me the possibility to carry out this project and gave me all the necessary material to make it possible. No matter the distance and time difference he has been helping me during all the process. Furthermore, Professor Ming-Jyh Chern admitted me into his laboratory providing all the necessary facilities for the research and making all possible to make me feel comfortable working there. The every week meeting has been totally useful to resolve any doubt and the group meetings were really interesting to learn about other projects. I would like to thank you both very much for your support and understanding over these months.

My time at National Taiwan University of Science and Technology has been highly productive and an extraordinary experience. Amber Xiao-Wen Huang and Jason Chao-Ching Kao kindly assisted me with any question, as well as, helping me with OpenFOAM and ICEM. I must also thank my colleagues of the laboratory for the friendly welcome.

I take this opportunity to express gratitude to Bhanu Prakash, who introduced me to OpenFOAM and was really helpful and patient at every difficulty I had when I was learning. He always tried to solve any problem when I was stuck, however busy he was.

I would also like to show gratitude to Emina Bakkali, the best co-worker I could find. We helped each other in the worst and best moments of the project, especially when we got blocked at some point. Without her, this experience would not be as fun as it has been.

And finally but not the least important, I am really thankful to my family who has made possible my exchange in Taiwan, and have always believed in me, as well as my friends, for being always there.

CHAPTER 1. INTRODUCTION

The study of two flow fluids has arisen in the recent years as a consequence of its presence in numerous engineering applications in space. The purpose of replacing single-phase for two-phase systems is due to the reduction in system mass and complexity leading to improvements in system reliability. This can be applied in the design of future fluid systems for spacecraft-impacting fluid bearing containers, thermal control system coolant reservoirs, water storage and management systems, liquid state low-gravity materials processing equipment, and biofluids handling instruments for in-flight human health systems, which is vital to further space exploration.

Resolving microgravity two-phase flow challenges are critical to help scientists and engineers to better understand its behaviour and be able to apply it on the numerous applications, since in the absence of gravity, the floatibility force has less effect, and therefore, the bubble generation is not constant and predecible. Another exciting area of study in microgravity is that of physical science. The propellants float inside the tanks and water droplets bounce off the recycling systems. This makes the design of fluid handling systems for spacecraft a difficult task. In addition, the study of two-phase flows in a microgravity environment can reveal behaviors that otherwise would be masked by the effects of gravity. Therefore the bubble generation in a microgravity environment is a crucial issue that needs an accurate control.

This research will focus on the mixture of gas and liquid to study the bubble generation process in a T-junction pipe. The geometry presented in Figure 1.1 consists of the union of two pipes in 90 degrees with two inlets, one for water and one for air, and the outlet with the mixture. Initially, the pipe will be full of water and the bubbles will start generating once the air enters the pipe, as a consequence of the capillary and the liquid drag forces, among others. We use numerical simulations due to the high cost of the experiments in spatial conditions, therefore thanks to CFD the two-phase modeling can be easily studied. In this study, OpenFOAM is the main tool for our simulations, as well as, Paraview and Matlab for the post-processing study.
Various studies have been done in this field with different configurations. This configuration has been studied before for [1, 2] but with the flow inverted, as presented in Figure 1.2. Figure 1.3 is another interesting case where another inlet for the water is implemented, a study and comparison between this case and ours can be found at [4] .

This report is organized, as first, a description of the methodology by an introduction to the CFD program, an explanation of the geometry and the mesh, a description of the fluids and the parameters of the study, and a specification of the boundary and initial conditions. In addition, a theoretical approach to the problem and an explanation of the calculation of the frequency, velocity and volume is given. Secondly, the validation chapter includes a description of the mesh convergence test with different velocity ranges of the air and the water, as well as a study of the bubble detachment and the contact angle. Thirdly, in results, a study of the bubble generation frequency f_B, the bubble volume V_B, the bubble velocity U_B, and finally the bubble length L_B is carried, as well as, a comparison with experimental values is done to see how reliable are the simulations. Finally, the conclusions are obtained followed by an introduction to possible

future work.

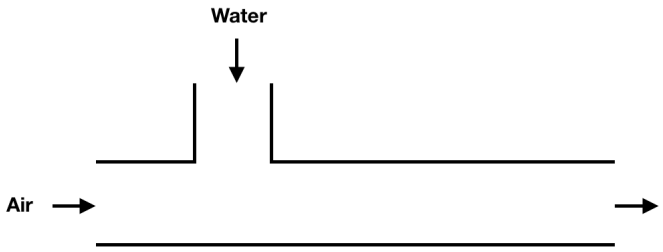

Figure 1.1: 2D Scheme of the case studied in this report

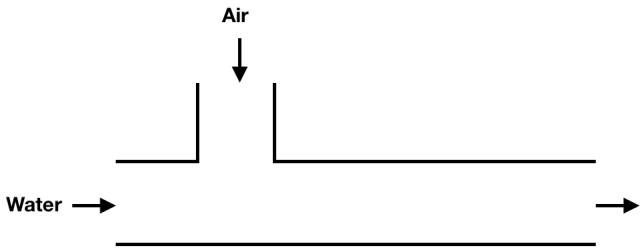

Figure 1.2: 2D Scheme of another case studied in [1]

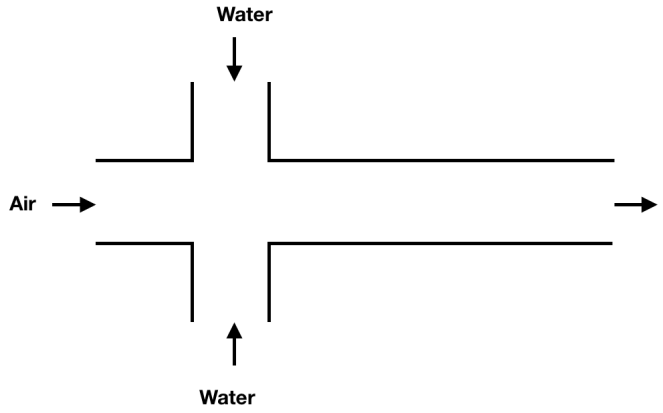

Figure 1.3: 2D Scheme of a double T-junction studied in [4]

CHAPTER 2. METHODOLOGY

2.1. CFD

Computational Fluid Dynamics (CFD), is a branch of fluid mechanics that uses numerical analysis and data structures to solve and analyze problems that involve fluid flows. Computers are used to perform the calculations required to simulate the interaction of liquids and gases with surfaces defined by boundary conditions.

Thanks to CFD, many engineering problems are covered, such as aerodynamics, gas turbines, turbo machinery, multiphase modeling, and so on, where some problems request experimental results which are often impossible to realize or economically non-viable. These are the most important reasons why CFD is of such important when dealing with fluid flow problems, it is cheaper and less time consuming. In recent years, it has become of common use both in the industry and the academic world, in order to work with fluid flow problems, due to its capability to show results faster and more accurately. However, validation and verification of the results is needed because it can give wrong results that look nice. Through validation and verification it is ensured that the results achieved are at least reliable.

CFD codes are based on numerical algorithms and consist of three main elements: pre-processing, processing and post-processing. In the pre-processing step, the inputs are given to the CFD software for the ongoing problem. It is based on the definition of the geometry, mesh generation, definition of fluid properties, the turbulence properties and of course the initial and boundary conditions. In the processing or solver part, the unknown flow variables are approximated by means of simple functions, then these approximations are introduced into the governing equations, they are discretized and then the algebraic equations are solved. Finally, in the post-processing step the data created from the previous step is analyzed, and the geometry, the grid, different vector plots, contour plots or surface plots can be visualized.

2.1.1. OpenFOAM

Open source Field Operation And Manipulation (OpenFOAM) is the program used to carry all the simulation in this project. This free and open-source software is really useful for the development of customized numerical solvers, and pre-/post-processing utilities for the solution of CFD.

The main advantages of using OpenFOAM are the free-license and the possibility to create individualized solutions that you specifically developed and product, unlike commercial software. On the contrary, at first, it might be difficult to understand since it does not have a visual interface. Therefore, Paraview is used to visualize the geometry and mesh and also for the post-processing of the results.

OpenFOAM uses a case folder structure to set-up and save case data. Figure 2.1 shows an initial state of a case. The main directories are: 0, constant and system.

The *0* directory contains the values for the initial and boundary conditions for the variables such as velocity, alpha, which is the volume fraction between air and water, and pressure. These values have to be defined for each section of the geometry: walls, inlets, and outlets.
The folder *constant* contains a full description of the case mesh in a subdirectory *polyMesh* and files specifying physical properties for the application concerned. The file *transportProperties* contains the material properties for each fluid (kinematic viscosity, density and surface tension) , *turbulenceProperties* contains the turbulence model that in our case is laminar, and finally the *g* file that defines the gravity value.
System folder is used for setting parameters associated with the solution procedure itself. It contains at least the following three files: *controlDict* where run control parameters are set including start/end time, time step and parameters for data output; *fvSchemes* where discretisation schemes used in the solution may be selected at run-time; and, *fvSolution* where the equation solvers, tolerances and other algorithm controls are set for the run. In addition there is the *decomposeParDict* file with the settings to divide the mesh over multiple processors in order to run the simulation on multiple processors and the *setFieldsDict* that initialize a volume of air.
In Appendix A, a wide explanation of openFoam can be found.

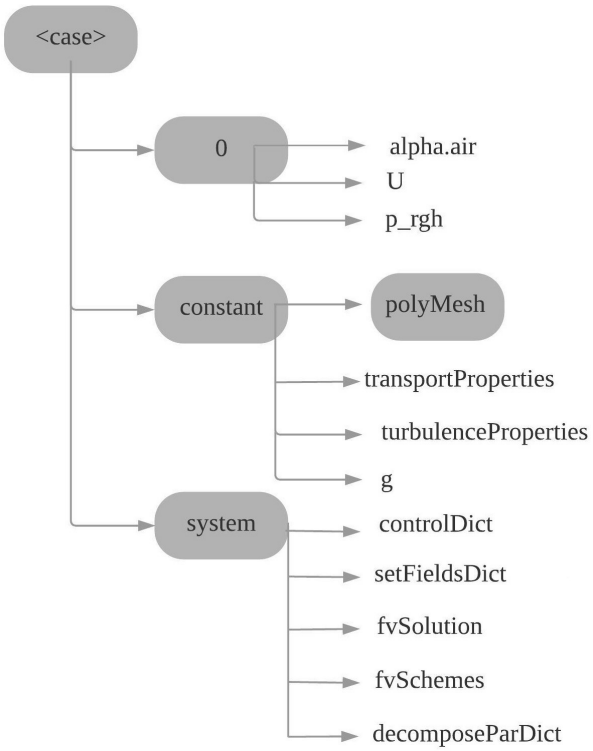

Figure 2.1: Scheme of the folder case for openFOAM

2.2. Pre-Processing

2.2.1. Geometry

The geometry in this study is based on the union of two cylinders in 90 degrees with the same diameter and different longitude, creating a T-junction pipe. The software used is OnShape, which is really easy to manage and modified afterward if some changes in the dimensions are needed. This program outputs STereoLithography (STL) files which can be read by various other CAD programs.
The inner diameter of the pipes is 1 mm, the longitude of the large one is 10 mm and 1 mm in the vertical pipe. This geometry is exactly the same as in [2] and [3] in order to make geometry comparisons. In Figure 2.2 a schematic drawing of the domain is given.

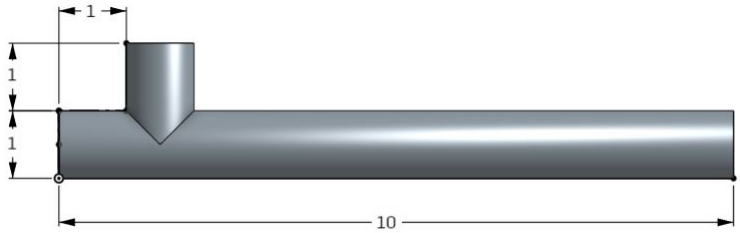

Figure 2.2: Lateral caption of the geometry with its dimensions in mm

OnShape does not allow to define the different surfaces, for that Salome was used. In Salome, it is possible to create groups to define the different parts of the geometry such as inlet1, inlet2, outlet, walls1, and walls2. Figure 2.3 shows the different surfaces defined, where inlet1 corresponds to the air inlet, inlet2 to the water inlet, the outlet corresponds to the exit of the bubbles and walls 1 and walls2 are the surfaces that define the walls of the pipes, vertically and horizontally, respectively.

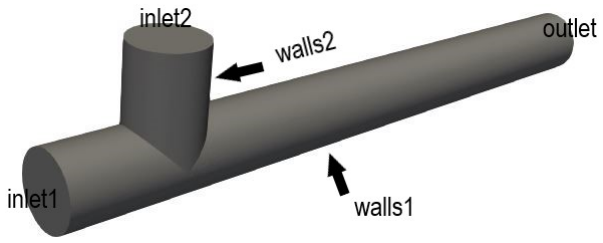

Figure 2.3: Caption of the geometry with its different surfaces: inlet1, inlet2, walls1 and walls2.

2.2.2. Mesh

Creating the mesh is an important process to obtain the desired results. Convergence tests are needed to verify if the accuracy and the structure of the mesh are sufficient, these tests are described in Section 3.

Many problems were presented in the process to select the best software to mesh since

the geometry is complicated, meshing is not straightforward. Salome, blockMesh, and snappyHexMesh were some of the programmes we tried. BlockMesh does not support complicate geometries like T-junction, and snappyHexMesh was more difficult to use. Salome is a good option for meshing despite is not easy to create the geometry. SimScale resulted to be a good option, but finally, ICEM was the one used. Table 2.1 presents the advantages and disadvantages of each programme.

Mesh software	Advantages	Disadvantages
blockMesh	Really convenient for simple meshes Is part of openFOAM	Does not support complex geometries Takes time: every parameter have to be entered manually
snappyHexMesh	Usefull for outside meshes (example: study the aerodynamic of a car) Is part of openFOAM	Difficult to use when the geometry is complex
Simscale	Online and free account for students Really easy to use	Does not allow to modify the parameters, too automatic
Salome	Free program Multiple options to generate meshes	Complicate to use, needs time to learn it
Ansys Icem	Useful to create complex meshes Quite easy to use	Not free

Table 2.1: Advantages and disadvantages of the different mesh programmes

ANSYS ICEM CFD is a popular proprietary software package used for CAD and mesh generation. The geometry can be imported directly from the STL file downloaded from Onshape. The mesh created is structured, in order to avoid possible problems. In a structured mesh, all interior vertices are topologically alike, while unstructured mesh does not follow a uniform pattern which can cause disturbances with the Courant number since its cells have not a uniform size. The Courant number (C_o) defined as $C_o = \frac{U_B \Delta t}{\Delta x}$, is an important parameter in the simulations since it affects to the accuracy and can imply convergence problems. It can describe the movement of the fluid depending on its value. When Co ≤ 1 the fluid particles move from one cell to another within one-time step; otherwise, it moves through two or more cells at each time step and this can negatively affect the convergence. Consequently, we will work with Co lower than 1 and since the velocity is a fixed value, an equilibrium between the time step and the cells should be done. For that, to have a reasonable time step we should avoid too little cells.

The total mesh is composed of 400 000 cells and different blocks which give more flexible than a single block due to the full control of the mesh grading, using edge meshing, with high-quality elements. The cell shape can be a triangle or a quadrilater, the choice depends on the problem and the solver capabilities, in this case, quadrilaters are chosen since the cell quality was better. In ICEM there is the option to calculate the cell quality based on the skewness which determines how close to ideal a face or cell is.
Figure 2.4 shows different zooms of the mesh, the inlets and outlet are separated into five blocks, a quadritaler in the center and in each edge a line splits the circumference to separate it in four more block, therefore it is easy to control the size of the cells and

you have more options to modify and vary the mesh. The other zoom corresponds to one of the crucial parts of the mesh since is the place where the two pipes intercept. In this part, the cells should converge uniformly without discrepancies to avoid problems with the simulations.

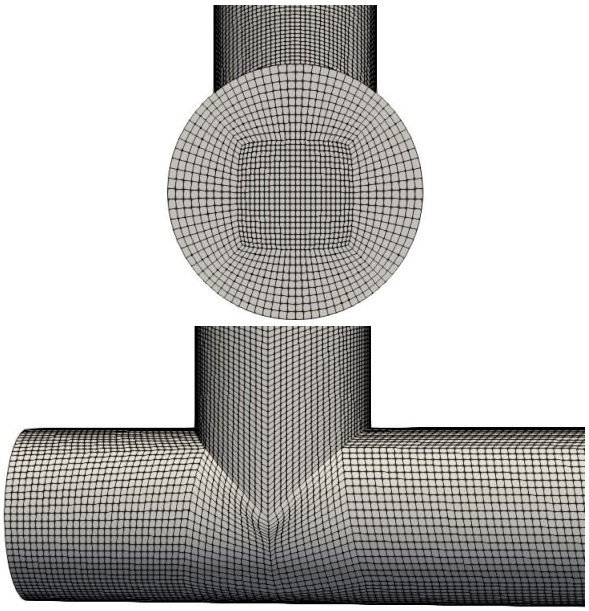

Figure 2.4: Frontal and lateral close-up view of the mesh

The mesh can be exported as a msh file and then easily converted to OpenFOAM by the command *fluent3DMeshToFoam*. This command will create the folder polyMesh inside constant, which contains the files with all the mesh information.

2.2.3. Fluids characteristics and work regime

The two-phase flow fluids studied in this project are water and air. Both fluids are considered incompressible and isothermal, at a room temperature of 25°C with a surface tension at the gas–liquid interface of σ=0.072 N/m. Its standard physical properties are: $\rho_L = 10^3$ kg/m^3, $\rho_G = 1.225$ kg/m^3, $\mu_L = 10^{-3}$ Pa·s and $\mu_G = 10^{-5}$ Pa·s.

The dimensionless numbers that characterize the problem are described in Table 2.2 for the range of velocities used in the results. The Bond number(B_o) defined as $B_o = \frac{\rho_L g \phi_C^2}{\sigma}$ was calculated in [2] with experimental results with the same velocity values as in this study, and the value was 0.139 so gravitational forces can be neglected ($B_0 \leq 0.29$ as studied in [16]). The Weber number is defined as $We = \frac{\rho_G U_B^2 \phi_C}{\sigma}$, the Reynolds number $R_e = \frac{\rho_L U_M \phi_C}{\mu_L}$ and the Capillary number $Ca = \frac{\mu_L U_{SL}}{\sigma}$. We can see in Table 2.2 that the $W_e < 2$ ([14]) which means that capillary forces overcome inertial forces, the $R_e < 2300$ ([15]) so it confirms we are working under a laminar conditions and finally the surface tensions effects are greater than liquid viscous forces ($C_a < 10^{-2}$).

	$U_{SL}[m/s]$	$U_{SG}[m/s]$	B_o	W_e	R_e	C_a
Minimum value	0.318	0.059	0.139	2.7×10^{-3}	377	4.4×10^{-3}
Maximum value	0.531	0.505	0.139	2.9×10^{-2}	1036	7.4×10^{-3}

Table 2.2: Characteristic dimensionless values

2.2.4. Initial conditions

For the inlets, the velocity for each entrance is imposed uniform and normal to the surface, with its corresponding value.
Alpha defines the fraction of water and air as one when is full of air. Therefore, in the air inlet, alpha is set to 1, and in the water inlet is set to 0. The gravity value is set to zero since, as we saw in Table 2.2, the Bond number is small, which means that the gravitational forces are smaller than the superficial forces, therefore, we can disregard the gravity. For that, the numerical results are in conditions similar to microgravity.

2.2.5. Boundary conditions

The walls have boundary type no-slip which means that the velocity is zero. This Dirichlet boundary condition describes that fluid near the boundary "sticks" to the wall, preventing fluid close to the wall from moving.
The inlets are freestream with its corresponding fluid velocity and the outlet has boundary type zerogradient, which means that there is no gradient of velocity there.
For the pressure, either the inlets and the walls are zero gradient and the outlet is set up as a pressure outlet, with a value of 101325 Pa.

2.3. Processing

2.3.1. Parallelization

In order to fasten the computational time, the simulations in OpenFoam run in parallel. To check how many cores we will use, test with 1, 2, 4, and 8 cores are done. The clockTime is the parameter to compare the results, the smallest value would be the best, for that we will do the test with the same conditions for 100 iterations.

Number of cores	Clock Time (s)
1	1507
2	880
4	580
8	1707

Table 2.3: ClockTime values for different cores

Table 2.3 presents the results, we can see that when the simulation is done with four cores the total time is the smaller one. Consequently, the simulations are executed with four cores.

2.3.2. Solver: InterFoam

The solver used is *interFoam* since it deals with multiphase problems. InterFoam is a two-phase solver for incompressible, isothermal and immiscible fluids using the VOF (volume of fluid) method.
It solves one momentum equation and one continuity equation which are the same for the two phases. The physical properties of one fluid are calculated as weighted averages based on the volume fraction of the two fluids in one cell.
The continuity and momentum equations take the form:

$$\nabla \cdot \mathbf{U} = 0 \tag{2.1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\rho \mathbf{U}) + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) - \nabla \cdot (\mu \nabla \mathbf{U}) = -\nabla p - \mathbf{F_s} \tag{2.2}$$

where F_s is the surface tension force that takes place only at the free surfaces.

The density, viscosity and velocity are calculated as:

$$\rho = \alpha \rho_G + (1-\alpha)\rho_L \tag{2.3}$$

$$\mu = \alpha \mu_G + (1-\alpha)\mu_L \tag{2.4}$$

$$\mathbf{U} = \alpha \mathbf{U}_{SG} + (1-\alpha)\mathbf{U}_{SL} \tag{2.5}$$

The values of α in a cell should range between 1 and 0. If the cell is completely filled with gas then $\alpha = 1$ and if it is filled with liquid then its value should be 0. At the interface, the value of α is between 0 and 1. The transport of α in time is expressed by an advection function:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{U}) = 0 \tag{2.6}$$

The surface tension force is calculated as:

$$\mathbf{F}_s = \sigma k \nabla \alpha \tag{2.7}$$

where k is the mean curvature of the free surface, determined from the expression:

$$k = \nabla \cdot \left(\frac{\nabla \alpha}{|\ \nabla \alpha\ |} \right) \tag{2.8}$$

2.4. Post-Processing

Once we have the results of the simulations, the post-processing is carried out with the open-source application Paraview, which allows to visualize the results from OpenFoam.

In addition, Matlab is used to calculate the different parameters and obtain the graphics. In Appendix A.3. an explanation of how to use Paraview can be found, and in C the Matlab codes are presented.
During the simulation, the value of alpha is stored for each time step. It would allow to calculate the parameters for the convergence tests and also for the results. These values do not correspond to all the geometry, it corresponds to the average of one slice normal to the pipe since we do not need to analyse the whole geometry, instead is more important a location near the outlet, when the bubble is more stable. The function 'faceSource' was used to save the average of alpha in a determined surface.
In order to calculate the parameters, we fix two monitors into the cross sections located at 7mm and 8mm in the x-direction. With one monitor we can extract the frequency of the bubble, its length and alpha. Nevertheless, another monitor is needed in order to calculate the velocity and the volume of the bubble.

Figure 2.5 shows the post-processing results of alpha as a function of time. From the graphic, we can extract the bubble period (Ts) and consequently we will have the frequency. In addition, Tf is the time the bubble needs to cross from the first monitor to the second, with this, we can obtain the velocity of the bubble.

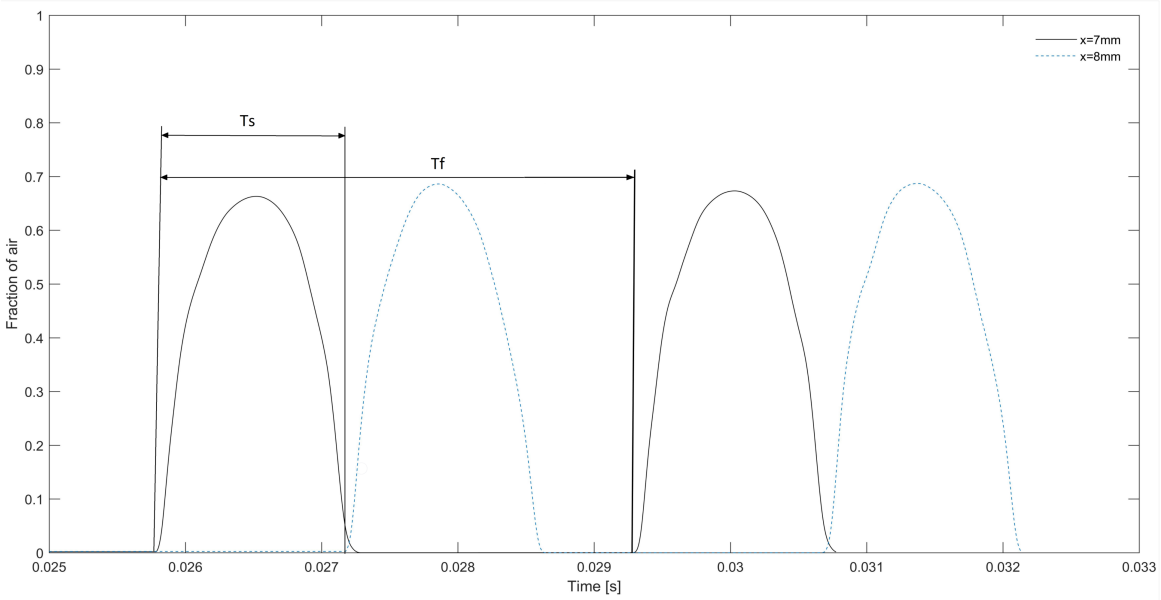

Figure 2.5: Post-processed results of alpha as a function of time, for $U_{SL} = 0.531$ m/s and $U_{SG} = 0.136$ m/s.

2.4.1. Bubble frequency

The frequency is easily calculated as the inverse of the period, and the period is the time between one bubble to the other.

$$f = \frac{1}{T_f} \tag{2.9}$$

2.4.2. Bubble velocity

The bubble velocity is defined as:

$$U_B = \frac{d}{T_s} \tag{2.10}$$

using the two monitors we can calculate the velocity as the distance between the monitors (1 mm) divided by the time needed for the bubble to travel this distance.

2.4.3. Bubble volume

The volume of the bubble is obtained by integrating the alpha over time, between entering the monitor (T_e) and leaving (T_l), and multiplying by its velocity and by the cross-section area.

$$\int_{T_e}^{T_l} \alpha dT \; U_B A \tag{2.11}$$

Where the $A = \pi\phi^2$

CHAPTER 3. VALIDATIONS

3.1. Mesh convergence tests

In order to obtain correct results, various tests are done to validate the convergence in the number of the cells of the mesh for different combinations of velocities. This is an important process before obtaining the final results to ensure accuracy in the final simulations. The whole procedure consists of multiple simulations, based on the combination of different meshes (Δx) for various combinations of superficial velocities for the gas and the liquid. For each simulation, the parameters of the frequency, the longitude and the velocity of the bubble are calculated to obtain the errors.

Three different meshes will be used: a coarse mesh called Mesh1 with 200 000 cells, the Mesh2 with 400 000 cells, and the finest one, Mesh3 with 600 000 cells, they were all created with ICEM. Mesh 3 is used to compare the results due to being the finest one should produce the bests results. The values of the number of cells for each mesh were selected based on the study in [2], where similar Δx were tested and validated.
At first, the idea was also to test the time step with Δt=$2.5 \times 10^{-6} s$, Δt=$5 \times 10^{-6} s$, and Δt=$10 \times 10^{-6} s$, but we needed to discard it. When executing the simulations with the smallest time step, it increased a lot the computational time. In addition, the biggest time step caused the simulation to stop, as a consequence of the Courant number, which increased its value. Therefore, the time step selected to conduct all the tests was Δt=$5 \times 10^{-6} s$, also used and validated before in [2], as the others values of Δ t that at first we wanted to use.
All the values of velocities and time steps selected in this study have been checked previously in other researches as in [2, 3]. Therefore, the range of velocities and the time step used are not randomly created, it has been validated before, and there is also experimental data to compare with.
Table 3.1 presents the different combinations of superficial liquid and gas velocities that are studied in the mesh convergence tests.

Combination	U_{SL} [m/s]	U_{SG} [m/s]
1	0.160	0.800
2	0.106	0.144
3	0.531	0.136
4	0.531	0.800
5	0.106	0.516
6	0.531	0.505

Table 3.1: Combinations of velocities proposed for the mesh convergence tests

Firstly, the combinations of velocities proposed were 1, 2, 3 and, 4, the objection was that the combinations 1 and 4 did not converge since the velocity of the air is higher than the liquid and the bubble generation is complicated. Consequently, two new combinations with better values were added (5 and 6).

In order to standardise the calculations, the procedure to obtain the values of the pa-

rameters will be the same. We will simulate until the bubbles converge and therefore, the frequency, length, and velocity of the bubble will be obtained by the mean of the last 6 bubbles. To validate these calculations, the standard deviation will be calculated to see how reliable are the means, by the formula:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \widetilde{x})^2}$$

where:
n is the number of bubbles used to calculate the mean (in this case 6).
x_i is the value of each bubble.
$\widetilde{x}$ is the value of the mean.

The main objective of the convergence tests is to validate a table with four combinations of velocities (2, 3, 5, and 6). Therefore we will be able to use any combination inside this ranges with the value of the number of cells and time step tested. Figure 3.1 shows the graphic with the velocities used.

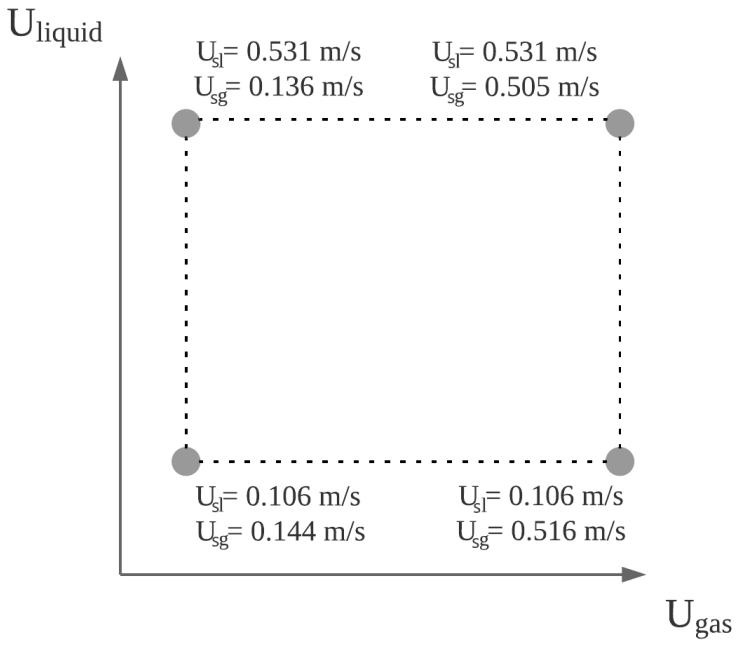

Figure 3.1: Graphic with the limits of the convergence tests

3.1.1. Combination1

As mentioned before, simulation on this combination without contact angle did not generate bubbles. For that, different contact angles were tested on Mesh3 to see which involve better results, and therefore, the convergence tests on the mesh can be made. The values tested were: 25°,30°, and 40°, these values were selected based on the study of [3].

Table 3.2 shows the results and standard deviations for the tests with contact angle. The standard deviation for the contact angle of 30° is the smallest in longitude and velocity

of the bubble. For the frequency, the standard deviation is smaller in the case of 40°, but comparing the results, the best combination is for the 30°. Therefore, new tests with the three meshes were done applying a contact angle of 30°, despite knowing that the results are not good.

Table 3.3 shows the values of frequency, length, velocity and its difference comparing to Mesh3. Mesh1 did not generate acceptable and constant bubbles since Δx is too large the mesh is not sufficiently accurate to obtain good results with OpenFoam. As a consequence, the parameters in Matlab cannot be calculated since there is no regular shape of the bubble to extract the results. The differences between Mesh3 and Mesh2 are too big and cannot be approved. Consequently, this test did not validate Mesh2. Furthermore, combination 1 is excluded since it does not generate valid results. This is due to the values of velocities since the liquid velocity is really low compared to the air velocity it cannot cut the bubble to help its generation. To obtain bubbles the liquid velocity must be bigger or the combination of both similar, if not, it is really difficult for the generation.

Figure 3.2 shows a 2D view of the bubble generation. It can be seen that the behaviour is unstable, different shape and size of the bubbles, therefore we will not use this combination to validate the mesh.

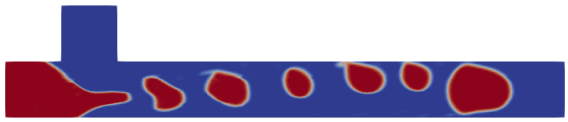

Figure 3.2: Bubble generation for $U_{SL} = 0.160$ m/s and $U_{SG} = 0.8$ m/s with a contact angle of 30°

3.1.2. Combination2

The values of the velocity of the air and the liquid in the combination 2 are quite similar, besides the liquid velocity is slightly smaller than the air velocity, it makes us expect the generation of the bubbles without problems.

Table 3.3 shows the values of frequency, length, velocity and its differences comparing to Mesh3. The differences between the Mesh3 and Mesh2 are less than 7%, the discrepancy in the length is bigger compared with the one in frequency and velocity, which are less than 3%. In Figure 3.3 we can see the graphics for the frequency, length and velocity for the number of bubbles. We can observe that all the values are really constant since the combination of velocities of the air and the water are quite similar. Furthermore, the behaviour between the finest and the medium mesh is more similar than with the one respect to the coarse mesh.

Table 3.4 presents the standard deviations for each mesh. We can clearly validate the way we proceed with the calculation since the deviations are really small, it can be neglected. Therefore, observing the values of the differences and the standard deviation, the Mesh2 can be validated for this combination of velocity.

For the previous tests, the bubbles attach to the walls. Therefore, in order to obtain rounded bubbles, the boundary condition of the contact angle is applied to see its af-

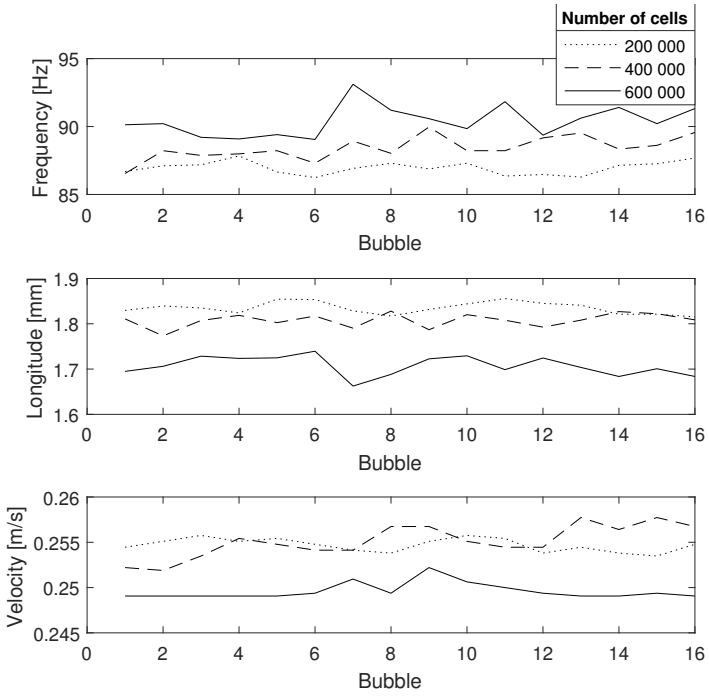

Figure 3.3: Graphic of the convergence tests for $U_{SL} = 0.106$ m/s and $U_{SG} = 0.144$ m/s and three different meshes

fectations. Table 3.2 shows the results when applying different contact angles. We can see that the values are really similar in all the cases because the contact angle for this combination does not help to detach the bubbles. Therefore, the results are really likely as in the case with no contact angle. Figure 3.4 perfectly shows the behaviour explained before and how the contact angle condition does not help on detaching the bubbles from the walls.

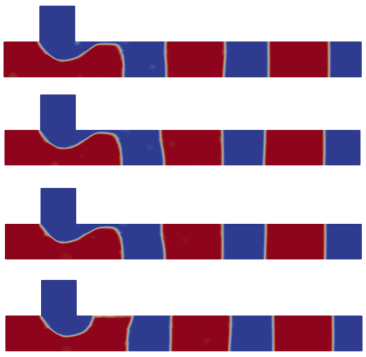

Figure 3.4: Bubble generation at the same moment for $U_{SL} = 0.160$ m/s and $U_{SG} = 0.8$ m/s and contact angles of 10°, 30°, and 90°

3.1.3. Combination3

Combination3 has the best combination of velocities for the air and the liquid since the value of the liquid velocity is bigger than the air and the difference between them is not really large. A problem with Mesh3 appeared in the tests. The results for the Mesh1 and Mesh2 were correct, but the Mesh3 did not diverge. This was caused by the Courant number, as the number of cells was bigger and the time step was the same as the other cases the Courant number did not bear such a big number of cells. In order to solve this problem, two solutions were possible, reduce the Δt, not a good solution since it would reduce the simulation speed and increase the simulation time, or decrease the number of cells creating a new mesh with 500 000 cells (Mesh4). The option selected was the second one.

Table 3.3 shows the discrepancies for the combination3. The differences between Mesh4 and Mesh2 are less than 3%. The difference in frequency is smaller for the Mesh1, but looking in the Figure 3.5, which shows the graphics for the frequency, length and velocity for the number of bubbles, we can observe that the values in the Mesh2 are more constant than the ones in Mesh1 and Mesh4. We can also see how the Mesh3 has really different results from the others because of its divergence. The behaviour of the three meshes is quite similar and constant. For the frequency and velocity, Mesh2 is the one that resembles more to Mesh4, otherwise, in length Mesh1 behave more like Mesh4. Overall, Mesh2 can be selected as the valid mesh for the results since its values are really constant and the differences with Mesh4 are insignificant.

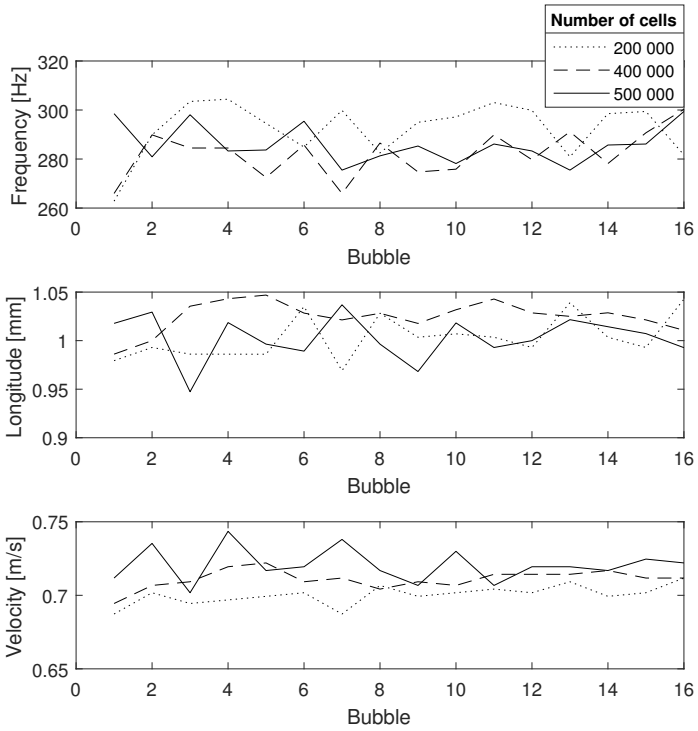

Figure 3.5: Graphic of the convergence tests for $U_{SL} = 0.531$ m/s and $U_{SG} = 0.136$ m/s and three different meshes

Table 3.4 presents the standard deviations for each mesh. We can clearly validate Mesh2

for this combination of velocities since the values on the deviation are also small. On the frequency the value is larger since the values of frequency are around 200 and 300, therefore this does not produce a large difference.

As in combination2, the bubbles attach to the walls, in this case only in the lower wall. Therefore, tests on the contact angle were also made to see if some improvements are made. Table 3.2 shows the results for the different contact angles. The most problematic cases were the angles 0° and 40° since there is more dispersion of air and it was difficult to calculate the initial and final points of the bubbles in Matlab. For 90° the standard deviation has the lower values, but in Figure 3.6 we can see the bubbles attaching to the lower wall, so it is not a good solution. The angles 0°,25, and 30° help the bubbles to detach from the walls, but we can see that there is also more gas dispersed (little bubbles). From 40°, bubbles start attaching to the wall again. In conclusion, the condition of the contact angle is dismissed since it does not produce the desired results and therefore a new condition should be used.

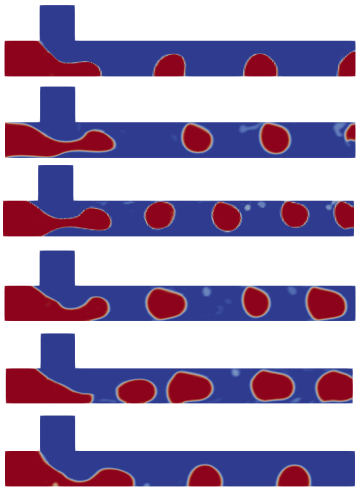

Figure 3.6: Bubble generation at the same moment for $U_{SL} = 0.531$ m/s and $U_{SG} = 0.136$ m/s for no contact angle condition, and contact angles of 0°, 25°, 30°, 40°, and 90°

3.1.4. Combination4

Test on the combination4 have also been made. As is combination1, without contact angle the bubble generation is not possible. Therefore, three different contact angles were tested to see its affectation.
Table 3.2 shows how the results of the different test on the contact angle and we can see how the values are the same for the 3 angles. This reaffirms that the contact angle has an strange behaviour and does not work well. For this combination, only the first five bubbles still have a normal behaviour. Therefore, to calculate the frequency, longitude and velocity of the bubble in this combination we take the last two bubbles since is not possible to take six bubbles as in the rest of the cases.

Table 3.3 shows the results for the different meshes applying a contact angle of 25°. The errors in the mesh with 200 000 cells are really big, and for the mesh with 400 000 cells

despite being smaller, are not acceptable. Therefore for this combination of velocities, reliable results cannot be obtained. Figure 3.7 shows the bubble generation with the mesh of 600 000 cells and the contact angle of 25°. The bubbles attach to the lower wall and do not have a normal shape.

Figure 3.7: Bubble generation for $U_{SL} = 0.531$ m/s and $U_{SG} = 0.8$ m/s

3.1.5. Combination5

This combination of velocity means to replace the combination 1 decreasing the gas superficial velocity to obtain better results.

Table 3.3 presents the results of the test for the meshes. As we can see, the differences between Mesh3 and Mesh2 are less than 2.5%. In addition, Table 3.4 shows the results of the standard deviation, and the values for the mesh with 400 000 cells are small and acceptable. Therefore, Mesh 2 is validated for this combination of velocities. In addition, in Figure 3.8 we can see that the behaviour between Mesh2 and Mesh3 is similar, while Mesh1 has more fluctuations and differs from the others.

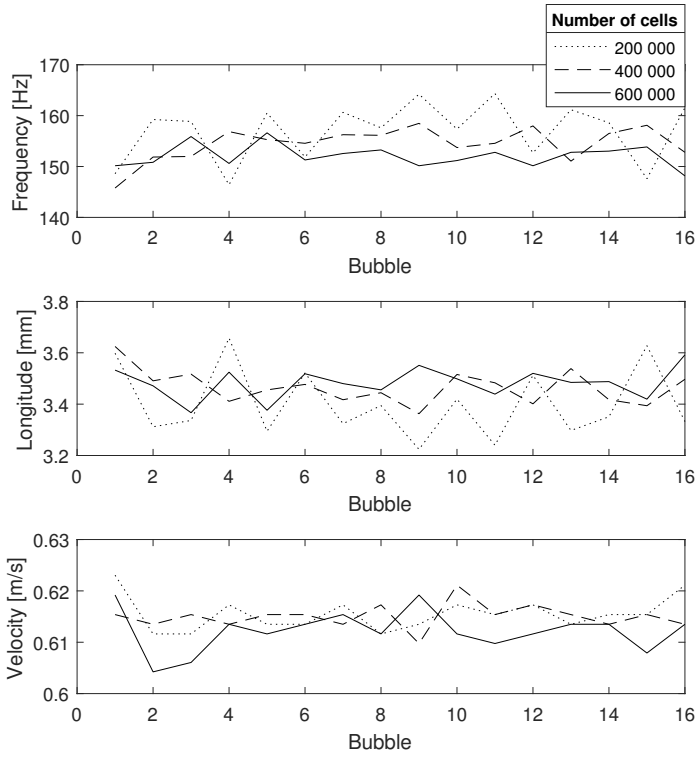

Figure 3.8: Graphic of the convergence tests for $U_{SL} = 0.106$ m/s and $U_{SG} = 0.516$ m/s and three different meshes

In Figure 3.9, the same behaviour of the bubbles attaching to the walls, as in combination 2 and 3, is presented. Since we saw that the contact angle did not help to detach the

bubbles from the walls, in this and next combination, tests on the contact angle were discarded.

Figure 3.9: Bubble generation for $U_{SL} = 0.106$ m/s and $U_{SG} = 0.516$ m/s

3.1.6. Combination6

This combination of velocity means to replace the combination 2 decreasing the gas superficial velocity, the same solution as in combination5.

Table 3.3 presents the results of the test for the meshes. Mesh2 has bigger differences than Mesh1 comparing to Mesh3, nevertheless in Figure 3.10 we can see that the values of frequency, length and velocity are more constant in Mesh2. In addition, Table 3.4 also confirms that Mesh2 has fewer fluctuations since the values of the standard deviation are smaller. Therefore, Mesh2 is validated.

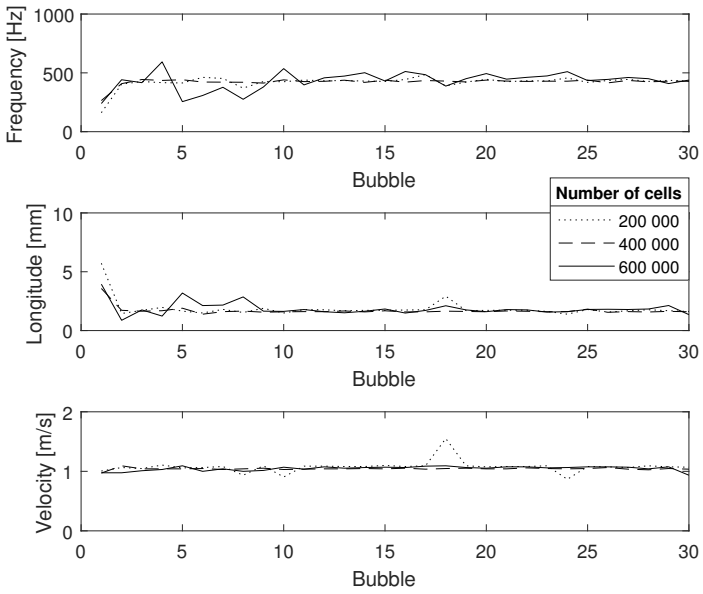

Figure 3.10: Graphic of the convergence tests for for $U_{SL} = 0.531$ m/s and $U_{SG} = 0.505$ m/s and three different meshes

Figure 3.11 shows how the bubbles attach to the lower wall, similar as in combination3.

Figure 3.11: Bubble generation for $U_{SL} = 0.531$ m/s and $U_{SG} = 0.505$ m/s

Tables 3.2, 3.3, and 3.4 present the values of all the tests of contact angle, tests on the mesh convergence and the values of the standard deviation for this section respectively. Comparing the values of the standard deviation for the cases 2 and 3 in Table 3.4, where the gas superficial velocity is quite similar, but the liquid superficial velocity is higher in the case 3, we can see how the standard deviation is bigger in the last case. Regarding the U_{SG}, if we compare the case 2 and 5, where the U_{SG} is bigger in the combination5 otherwise the U_{SL} is quite similar, we observe that the standard deviation increases as well. Therefore, we can conclude that the standard deviation increases when the gas and liquid superficial velocities increase.

Comb.	Contact Angle(°)	f [Hz]	σ_f [Hz]	L_B [mm]	σ_{L_B} [mm]	U_B [m/s]	σ_{U_B} [m/s]
	25	97.03	71.05	0.250	0.214	0.014	0.004
1	30	118.36	51.05	0.496	0.174	0.041	0.001
	40	40.14	37.78	0.534	0.256	0.534	0.966
	10	88.06	0.59	1.807	0.012	0.254	1.807
2	30	89.27	1.13	1.824	0.013	0.258	0.001
	90	88.69	0.61	1.811	0.013	0.256	0.001
	0	333.24	39.73	1.492	0.342	0.854	0.012
	25	469.71	75.19	0.718	0.787	0.499	0.645
3	30	464.91	74.65	0.185	0.086	0.145	0.044
	40	493.93	84.98	1.635	0.508	1.014	0.087
	90	291.09	6.39	1.023	0.009	0.719	0.003
	10	425.72	87.80	1.703	1.704	0.874	0.940
4	25	425.72	87.80	1.703	1.704	0.874	0.940
	40	425.72	87.80	1.703	1.704	0.874	0.940

Table 3.2: Study of the influence of the contact angle boundary condition on three independent parameters for different combinations of velocities with the Mesh3, in the combination3 with the Mesh4

3.2. Bubble detachment

The detachment of the bubbles from the wall is crucial in order to obtain reliable results. Since we saw that the contact angle condition itself does not produce the desired results, different conditions have been tested in order to see which presents the best results.

Regarding the velocity in the walls, two different boundary conditions have been tested to see which adapted better: slip and non-slip. Slip condition defines that the velocity in the walls should be different than 0, and non-slip imposes zero velocity in the walls. In addition, the condition of wettability is imposed by defining $\alpha = 0$ in the walls, this is a crucial condition to avoid the bubbles to attach to the walls.

First, two test were made, one with the slip condition and the other with the no-slip condition. Table 3.5 shows how the standard deviation for the slip condition is bigger, in addition analysing the animations in Paraview we could see that the bubbles were not stable, and they moved from the lower to the upper wall. Therefore, the slip condition was dismissed.

The condition of no-slip is the one selected, nonetheless it stills needs to add a boundary layer to the mesh in order to obtain accurate results near the wall since the bubbles scrape

Comb.	Number of cells	f [Hz]	E_F [%]	L_B [mm]	E_{L_B} [%]	U_B [m/s]	E_{U_B} [%]
1	600 000	106.08	-	6.606	-	0.237	-
	400 000	45.25	57.3	31.644	378.9	0.428	80.953
	200 000	-	-	-	-	-	-
2	600 000	90.79	-	1.699	-	0.249	-
	400 000	88.91	2.1	1.811	6.6	0.256	2.8
	200 000	86.87	4.3	1.833	7.9	0.254	1.9
3	600 000	351.09	-	0.645	-	0.369	-
	500 000	286.02	-	1.006	-	0.718	-
	400 000	288.31	0.8	1.026	2.1	0.714	0.6
	200 000	293.89	2.8	1.012	0.8	0.705	1.9
4	600 000	397.36	-	0.335	-	0.728	-
	400 000	373.18	6.1	0.577	41.5	0.228	68.7
	200 000	133.19	66.5	67.706	394.1	0.128	82.5
5	600 000	151.79	-	3.491	-	0.612	-
	400 000	155.16	2.2	3.455	1	0.615	0.6
	200 000	157.65	3.9	3.393	2.8	0.616	0.8
6	600 000	456.84	-	1.744	-	1.067	-
	400 000	429.14	6.1	1.635	6.3	1.045	2.1
	200 000	434.57	4.9	1.764	1.2	1.097	2.8

Table 3.3: Study of mesh convergence for three meshes and three independent parameters with six different combinations of velocities

Comb.	Number of cells	σ_f [Hz]	$\overline{\sigma_f}$	σ_L [mm]	$\overline{\sigma_L}$	σ_{U_B} [m/s]	$\overline{\sigma_{U_B}}$
2	600 000	0.91	0.01	0.015	$8{,}83{\cdot}10^{-3}$	$3.639{\cdot}10^{-4}$	$1.46{\cdot}10^{-3}$
	400 000	0.59	$6.64{\cdot}10^{-3}$	0.012	$6.63{\cdot}10^{-3}$	0.002	$7.81{\cdot}10^{-3}$
	200 000	0.58	$6.68{\cdot}10^{-3}$	0.016	$8.73{\cdot}10^{-3}$	$7.310{\cdot}10^{-4}$	$2.88{\cdot}10^{-3}$
3	600 000	73.02	0.208	0.612	0.949	0.197	0.534
	500 000	7.72	0.027	0.012	0.012	0.006	$8.36{\cdot}10^{-3}$
	400 000	8.20	0.028	0.011	0.011	0.002	$2.8{\cdot}10^{-3}$
	200 000	9.88	0.034	0.023	0.023	0.005	$7.09{\cdot}10^{-3}$
5	600 000	2.18	0.014	0.062	0.018	0.002	$3.27{\cdot}10^{-3}$
	400 000	2.87	0.019	0.059	0.017	0.001	$1.63{\cdot}10^{-3}$
	200 000	6.31	0.040	0.147	0.043	0.003	$4.87{\cdot}10^{-3}$
6	600 000	33.67	0.074	0.169	0.097	0.013	$1.22{\cdot}10^{-3}$
	400 000	8.51	0.019	0.096	0.059	0.013	$1.24{\cdot}10^{-3}$
	200 000	13.04	0.030	0.147	0.083	0.087	$7.93{\cdot}10^{-3}$

Table 3.4: Study of the standard deviation for three meshes and three independent parameters with six different combinations of velocities

the upper wall. Different boundary layers have been created to see which presents best results. Case1 is created by adding boundary layers and deleting cells in the center to avoid a mesh with more than 500 000 cells. The problem was that is was too coarse. Therefore, Case2 was created with more boundary layers, and the results were better. In the animations, the contour can still be seen unfocused, but the results are good. In order to try to refine this boundary, a new Case3 was created with more cells in the center, but since this increased a lot the total number of cells, the number of boundary layers were decreased.

Table 3.5 shows the results for the different boundary layers. Case2 has the smallest

standard deviations, which means fewer fluctuations, so the bubbles are more constant.

	f [Hz]	σ_f [Hz]	L [mm]	σ_L [mm]	U_B [m/s]	σ_{U_B} [m/s]
Slip	409.55	136.41	0.892	0.438	0.845	0.161
NoSlip	335.67	130.99	0.229	0.095	0.089	0.010
Case1	309.94	126.19	1.087	0.108	0.783	0.022
Case2	357.48	26.76	1.023	0.033	0.837	0.016
Case3	312.45	118.36	1.019	0.089	0.793	0.046

Table 3.5: Study of different conditions and three independent parameters for $U_{SL} = 0.531$ m/s and $U_{SG} = 0.136$ m/s

In Figure 3.12 we can see a caption with all the case and the most similar to the experimental data is clearly the Case2.

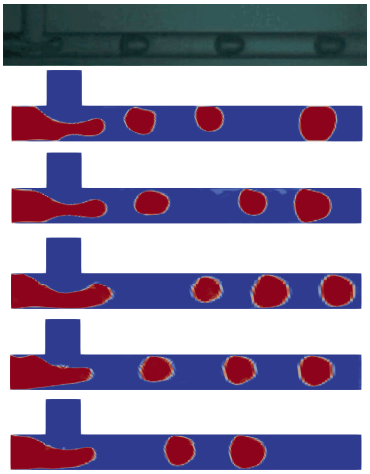

Figure 3.12: Bubble generation at the same moment for experimental data, slip condition, no-slip condition, Case1, Case2, and Case3

Consequently, the best case is with the boundary conditions of no-slip, wettability on the walls and using boundary layer as Case2. This will be the conditions used for the final simulations in the Results section.

3.2.1. Contact angle

The contact angle is the angle between a liquid-vapor interface and a solid surface. It quantifies the wettability of a solid surface. A given system of solid, liquid, and vapor at a given temperature and pressure has a unique equilibrium contact angle. However, since we do not know which contact angle has in reality this problem, we should try different values to see which match better the experimental results. For contact angles greater than or equal to 90 ° it is said that the adhesive does not wet the substrate, it does not generate any adhesion between the adhesive and the substrate; if the contact angle is below 90 degrees we say that the adhesive wets the substrate causing adhesion between both materials.

Figure 3.13 shows how the angle is defined in OpenFoam. The angle normally is defined by the other side, for that when defining the value in OpenFoam it should be 180-the

value you want. In Appendix A.1.4. an explanation of how to define this condition in OpenFoam is described.

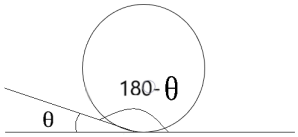

Figure 3.13: Contact angle definition in OpenFoam

The contact angle boundary condition has also been tested in addition to the wettability condition in order to see if then the results improve. Three different values have been applied to see the variations: 0º, 45º, and 90º. Table 3.6 shows the results of frequency, longitude, velocity and volume for the different contact angles with the case selected in the previous section. We can see that when we apply the contact angle condition the results are the same for the all the values. This happened before in the combination2. In addition, it also shows the results of the standard deviation, and the values are really similar when applying contact angle and when we do no apply it.

Consequently, the test in the Results section are carried out without applying contact angle, since we cannot see a clearly difference in Figure 3.14 and also the values of frequency, length, velocity and volume of the bubble does not became better. In conclusion, we have not been able to understand how the contact angle works in OpenFOAM, further work on this subject is required.

Contact angle	f [Hz]	σ_f [Hz]	L_B [mm]	σ_{L_B} [mm]	U_B [m/s]	σ_{U_B} [m/s]	V_B [10^{-9}m^3]	σ_{V_B} [10^{-9}m^3]
No angle	357.80	37.51	1.024	0.011	0.836	0.007	7.764	0.191
0º	342.69	37.26	1.029	0.026	0.836	0.006	7.712	0.350
45º	342.69	37.26	1.029	0.026	0.836	0.006	7.712	0.350
90º	342.69	37.26	1.029	0.026	0.836	0.006	7.712	0.350

Table 3.6: Study of the influence of the contact angle boundary condition on three independent parameters and its standard deviations for $U_{SL} = 0.531$ m/s and $U_{SG} = 0.136$ m/s

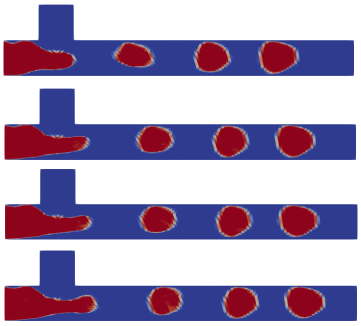

Figure 3.14: Bubble generation at the same moment for no contact angle condition, and contact angles of 0º, 45º, and 90º

CHAPTER 4. RESULTS AND DISCUSSION

In this chapter, an analysis and discussion of the results is done. Numerical simulations are compared with experimental data, for a total of 13 tests. The bubble frequency, velocity, volume and length are the parameters used to validate the results. We will use three different groups of velocities, each group with the same liquid velocity to study how the gas superficial velocity affects the parameters. These velocities are not selected randomly, the combinations have been tested before, and the experimental data we use to compare with our results have exactly the same values of liquid and gas velocity. All simulations are carried out with a mesh of 400 000 cells, Δt=5$\times10^{-6}$s, no-slip and wettability condition, and without applying the contact angle condition, as explained in Section 3. Tables B.1 and B.2 in Appendix show all the combinations of velocities and its results.

Figures 4.1 and 4.2 show the bubble generation comparative between experiments and simulations. The first two columns correspond to U_{SL}=0.106 m/s (ref 1 to 3, see Table B.1, from above to below), the group below corresponds to U_{SL}=0.318 m/s (ref 4-8) and the last group corresponds to U_{SL}=0.531 m/s (ref 9-13). We can see that depending on the liquid and gas superficial velocities, the bubble shape changes , being more rounded when the velocity of the liquid is higher than the gas. In addition, when the liquid superficial increases, the bubble frequency increases and more bubbles are generated, as we can see in the group with U_{SL}=0.531 m/s. When the gas superficial increases the bubble length also increases and the bubble does not fit in the capillary, which causes the large shape of the bubble. When both liquid and gas superficial velocities are slow, as in the case of U_{SL}=0.106 m/s (Figure 4.1), the differences between the three cases are almost imperceptible.

In general, the bubble generation between experiments and simulations is really similar, bubbles were generated with high regularity and small size dispersion in the experiments. In order to see how regular are the simulations, the calculation of the normalized standard deviation of the frequency, which is defined as the division between the standard deviation and the frequency, is done and presented in Table B.1. If the value is similar or inferior to 1$\times10^{-2}$ then we can say that the generation is regular, otherwise, it is not. There are some simulations where the generation is not regular, such as in combination 9, and in the case of combination 10 is on the limit. Overall, the numerical tests present regularity in the generation and small size dispersion of the bubble. Moreover, the error bars are added in the Figures 4.3, 4.4, 4.5, and 4.6 in order to see how accurate are the calculations of the parameters, as well as, adds precision to the results.

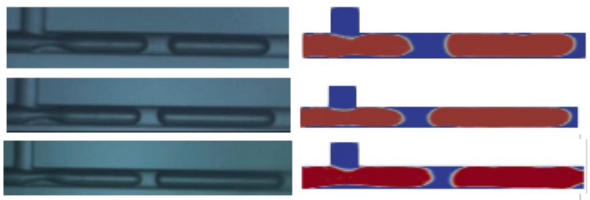

Figure 4.1: Bubble generation comparison between experimental results (1st column) and simulations (2nd column) for the combinations from 1 to 3 (from top to bottom of the figure)

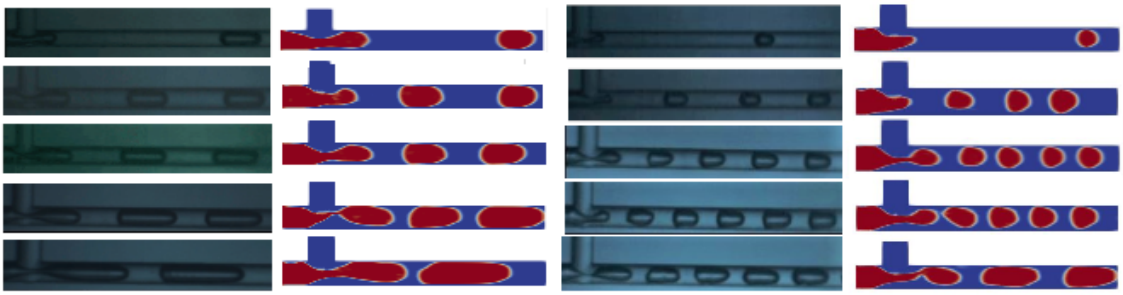

Figure 4.2: Bubble generation comparison between experimental results (1st and 3rd columns) and simulations (2nd and 4th columns) for the combinations from 4 to 13. U_{SG} increases when lowering in the figure, and U_{SL} increases while going to the right.

4.1. Bubble frequency

The bubble frequency has been studied for different liquid superficial velocities, focusing on the evolution when increasing the gas superficial velocity. Figure 4.3 shows the results, empty symbols correspond to experimental data and solid symbols to numerical simulations. We can see two different behaviours for the frequency depending on the gas superficial velocity as [2] explained before. For very low gas flow rate values, it follows a linear tendency that progressively curves until arriving a saturation value.

This behaviour is also reproduced in the graphic, continuous lines correspond to experimental data and discontinuous lines with simulated data, characterised by [2] as:

$$f = f_{sat}(1 - e^{\frac{-a_0}{f_{sat}}U_{SG}}) \tag{4.1}$$

Where a_0 is the initial slope of the linear regime and f_{sat} is the saturation frequency, these values are presented in Table 4.1 for experimental and simulated data. For the cases with U_{SL}=0.106, 0.318 m/s, the values of a_0 and f_{sat} for experiments and simulations are of the same order of magnitude but larger for the simulations. In the case of U_{SL}=0.531 m/s, the simulations do not follow the same behaviour due to the divergence of the last two points.

$U_{SL}[m/s]$	Experiments		Simulations	
	$a_0[m^{-1}]$	$f_{sat}[1/s]$	$a0[m^{-1}]$	$f_{sat}[1/s]$
0.106	675	97.51	417.3	108.9
0.318	859.9	270.8	1118	529.8
0.531	1467	1151	5088	479.5

Table 4.1: Values of a_0 and f_{sat} obtained when fitting the experimental and numerical data by using Eq. 4.1

Numerical data follows the same behaviour as experimental data but quantitatively bigger. The errors oscillate between 1.1% and 141.2%, with an average of 48.1% and a standard deviation from 3.3 Hz to 88.5 Hz. We are conscious that these errors are too large quantitatively, for that, future work is necessary in order to obtain the appropriate boundary conditions and achieve better results. Nonetheless, qualitatively it agrees with experimental data.

There appear big discrepancies in the cases of U_{SL}=0.531 m/s and U_{SG}=0.352, and 0.505 m/s where the frequency does not follow the expected behaviour. These simulations should be dismissed since they are not credible, but we did not find out what is the cause of the discrepancies, therefore, more future work is necessary. Different to the rest of combinations, the case of U_{SG}= 0.352 m/s presented numerous fluctuations in the frequency during the bubble generation and depending on how many bubbles we take to calculate the statistic the values changed crucially. To obtain the value of frequency we took the first 16 bubbles and calculate the mean of the last six, this was the case with fewer fluctuations, if we take 30, 60 or 100 bubbles the standard deviation increases considerably.
We can see that as the liquid superficial velocity increases, the frequency increases too, simulations with U_{SL}=0.531 m/s (squares) have bigger values than the other groups. In addition, the standard deviation becomes also bigger, that the liquid superficial velocity makes the frequency less stable, with more fluctuations. When increasing the gas superficial velocities, the frequency also increases.

In addition, the standard deviation also becomes larger, so we could say that the liquid superficial velocity can cause instabilities to the frequency, more fluctuations. Although the fluctuations are not only related to the standard deviation, but also depend on the relationship between the standard deviation and the average value of the frequency.

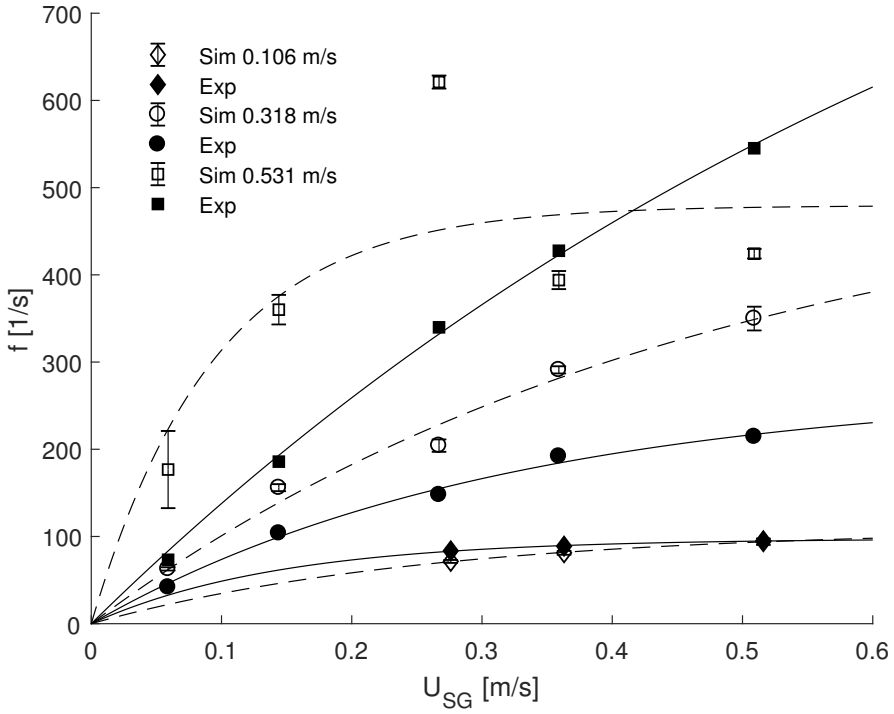

Figure 4.3: Bubble frequency as a function of the superficial gas velocity for three different liquid superficial velocity. Continuous lines correspond to the experimental fittings and discontinuous lines to the simulation fittings.

4.2. Bubble volume

To study the bubble volume it has been adimensionalize with the cross-section area and the capillary diameter. Figure 4.4 shows the graphic of the dimensionless volume as a function of $U_{SG}/f\phi_c$ which is obtained by normalising the expression $Q_G = V_B f$ with the cross-section area times the capillary diameter to obtain a non-dimensional expression of the bubble volume, explained in [2].
The function y=x is also plotted as a theoretical prediction in order to see if the results fit this prediction. We can see how simulated and experimental values follows the linear behaviour and are really similar qualitatively, increasing the bubble volume when increasing the gas superficial velocity. The bubble volume in the simulations with U_{SL}=0.318 m/s and 0.531 m/s is smaller than the experiments due to the larger values in the frequency, which causes the volume of the bubble to be smaller. For the case of U_{SL}=0.106 m/s depends on the combination.
The standard deviation for the simulations goes from 0.012 to 0.343$\times 10^{-9}$ m^3, and in Table B.1 we can observe the normalized value of the standard deviation for the volume, where most of the cases are inside the range of regularity despite some cases such as combinations 6, 8,and 9. The errors between experimental and simulated data move from 0.3% to 48.3% with an average of 30.1%. The values of volume for the combinations with U_{SL}=0.106 m/s are almost equal for experiments and simulations, but since the frequency is quite smaller in the simulations, we can see in the graphic how the simulated points are diverted to the right. In addition, there is a point which corresponds to the combination U_{SL}= 0.318 m/s and U_{SG}= 0.267 m/s that moves away from the linear characterisation.

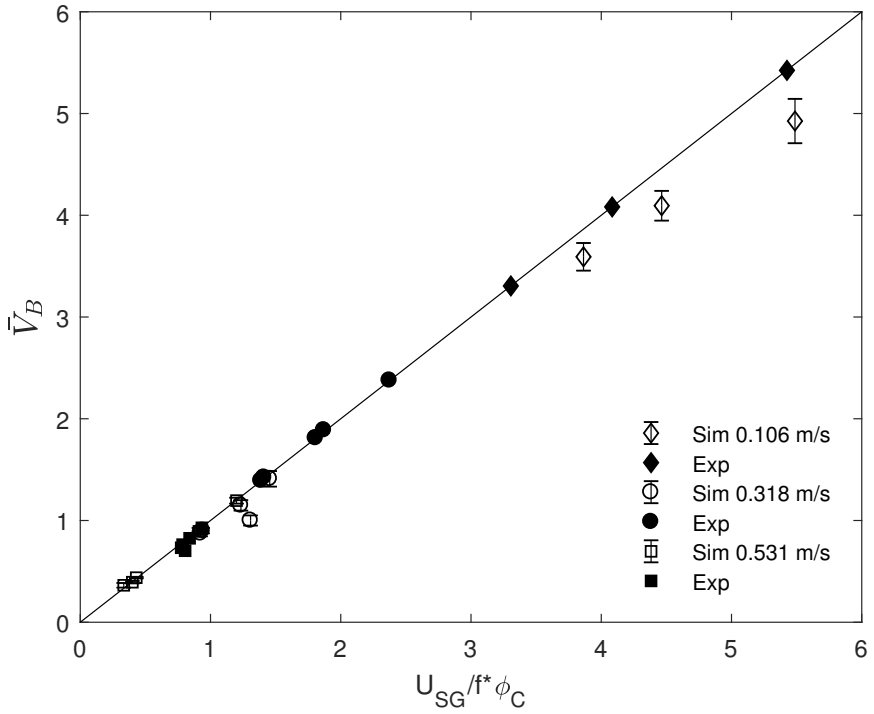

Figure 4.4: Normalized bubble volume as a function of $\frac{U_{SG}}{f*\phi_C}$ for three different liquid superficial velocity

4.3. Bubble velocity

The bubble velocity must be proportional to the mixture velocity (U_M), as the drift-flux model [19] predicts for cases under non-dominant gravitational forces, so it can be characterised as $U_B = C_0 U_M$, where C_0 is the void fraction distribution coefficient, explained in [2]. Figure 4.5 shows the bubble velocity as a function of the mixture velocity. It also illustrates the linear tendency for experimental (continuous lines) and for simulated (dashed lines) data where the value of C_0 has been calculated and the results are 1.13 and 1.168 respectively. The difference is only 3.4%, proving that simulations and the experiments are really similar qualitatively, as well as they follow perfectly the linear behaviour. Since the values of C_0 are greater than one it means that bubbles move faster than the mixture superficial velocity, and they are coherent with the values reported in [2].
The standard deviation is almost negligible, as we can see in the figure that the error bar is really short, with a minimum of 0.004 m/s and a maximum of 0.043 m/s. Moreover, in Table B.2, we can see the normalized value of the standard deviation, where all the cases are around 1×10^{-2}, which means that are regular. Regarding the errors, the average is 8.8% with values between 2.6% and 20.8%. In the case with the larger liquid superficial velocity U_{SL}=0.531 m/s, the velocity in the simulations is always bigger than the experiments. Otherwise, when U_{SL}=0.106 m/s and U_{SL}=0.318 m/s the values are smaller.

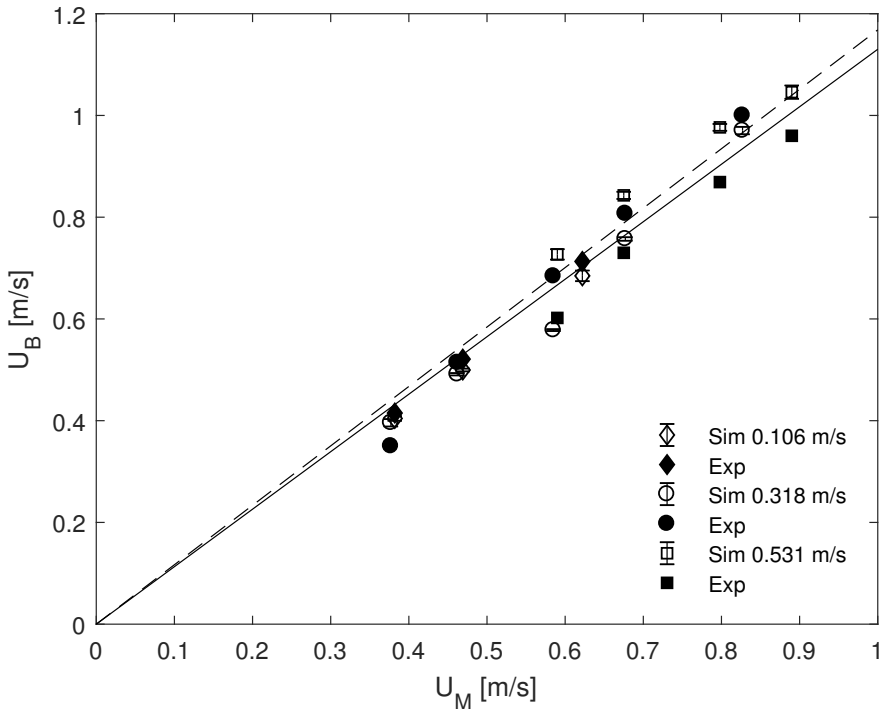

Figure 4.5: Bubble velocity as a function of the mixture superficial velocity for three different liquid superficial velocity. Continuous line corresponds to the experimental fitting and discontinuous line to the simulation fitting.

4.4. Bubble length

The bubble length can be considered as the gas displacement during the time needed for a bubble to be generated since the bubbles occupied nearly all the capillary the gas displacement can be assumed as the gas superficial velocity. Therefore, this length normalized with the capillary diameter is $L^* = \frac{U_{SG}}{f\ \phi_c}$. In Figure 4.6 the bubble length has been normalised with the capillary diameter and plotted as a function of $\frac{U_{SG}}{f\ \phi_c}$. The function used to fit the data, predicted by [3], is:

$$L = C1 + C2\frac{U_{SG}}{f\phi_C} \tag{4.2}$$

Where the fitting constants C1 and C2 have been calculated from experimental and simulated data, and the results are: C1=0.34, C2=1.2 (experiments, continuous line) and C1=0.54, C2=1.107 (simulations, dashed line). We can see in Figure 4.6 that the two fittings are really similar, as well as, simulations and experiments have similar behaviour, the bubble length increases when increasing the bubble velocity. Simulations with smaller liquid superficial velocity have bigger values of longitude because the frequency is smaller, we have seen this before in Figure 4.1.
The errors have a minimum value of 0.5% and a maximum of 47.1%, with an average of 22.7%. The standard deviation varies between 0.018 mm and 0.526 mm, which is totally acceptable. In addition, the same as in the case of the bubble velocity, for the longitude, the normalized values of the standard deviation, presented in Table B.2, follows the regular tendency.

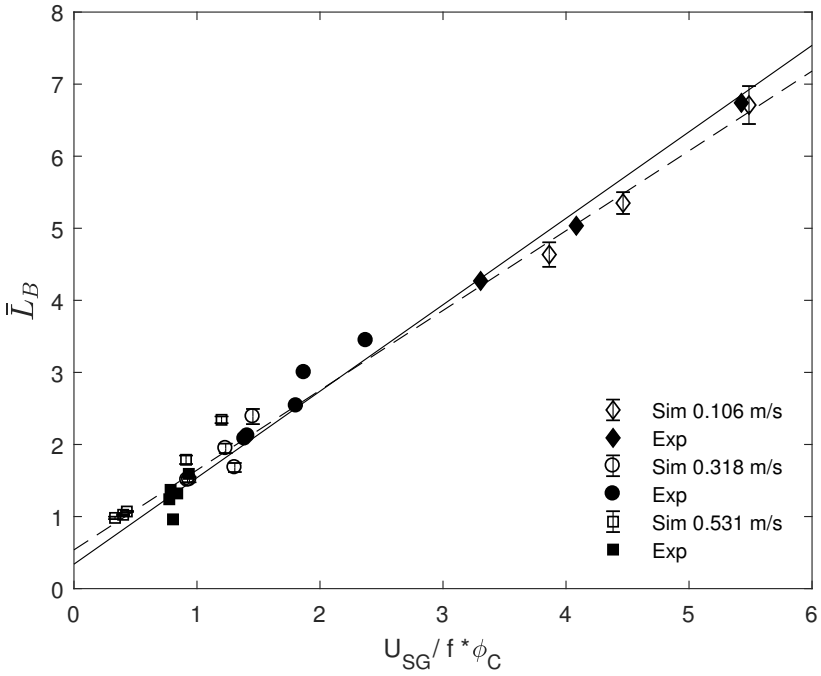

Figure 4.6: Normalized bubble length as a function of $\frac{U_{SG}}{f*\phi_C}$ for three different liquid superficial velocity. Continuous line corresponds to the experimental fittings and discontinuous line to the simulation fitting.

CHAPTER 5. CONCLUSIONS

This paper presents the study of the bubble generation process in a capillary T-junction under a microgravity environment. The mixture is composed of air and water, and OpenFOAM v2.0 is the CFD program used to perform the simulations. Numerical results are compared with experimental data, with both cases following the same conditions.

First of all, validations on the mesh were done to assure accuracy in the results, as well as, a selection of the best boundary conditions to avoid the bubbles attach to the walls. This process is critical to obtain reliable and coherent results. We obtained that the mesh with 400 000 cells was suitable to perform the simulations, due to its small fluctuations and similarity to a finer mesh. The wettability condition was selected to detach the bubbles from the walls, in addition to adding boundary layers to the mesh in order to obtain more precision near the walls. The contact angle condition was also analysed but the results were not satisfactory, leading to confusion on how does this condition works in OpenFOAM.

After the validations were made, the final simulations were carried out for 13 different combinations of gas and liquid superficial velocities, inside the range previously tested and with the same values as the experimental data. The parameters used to compare and analyse were the bubble frequency, volume, velocity, and length. Theoretical fittings were also added to the study. We obtained that simulations and experiments are really similarly qualitatively, with some differences quantitatively, most of them in the frequency. In order to improve the results, further research is needed in order to obtain the best boundary conditions, as well as, understand the behaviour of the contact angle in OpenFOAM, to lead into better results and avoid the discrepancies encountered in this study. Nevertheless, our results can be acceptable since the behaviour follows the expectations. There are only two combinations of velocities U_{SL}=0.531 m/s and U_{SG}=0.352, 0.505 m/s that diverge from the expected behaviour in the frequency, as it can be seen in Figure 4.3, but overall, the results in the other parameters such as the velocity, longitude, and volume are correct. In addition, numerical simulations satisfactorily reproduced the formation of bubbles, as Figures 4.1 and 4.2 demonstrate, both in the stabilised shape and in the evolution of shape during its formation. We also observe the direct relationship between all the studied parameters, there is a physical connection which leads to a dependence.
In summary, by analysing all the data, we deduced that there is no regular pattern to say which combinations of velocities are the best, since depending on the parameter studying one combination is better than the other. Focusing on the normalized standard deviation of the bubble frequency, volume, velocity, and longitude, which defines if the parameter is regular or not, we can see that most of the simulations are regular. Nevertheless, it does not follow a rule of how the liquid and gas superficial velocities affect the normalized standard deviation, it changes randomly, but always maintaining the same order, there are no critical changes, in exception of the combination 9.

As a future work, an extensive study of the contact angle condition should be done to understand its behaviour and further improve the results. In addition, if new simulations with a lower time step and a fine mesh are performed, the values would be more precise and results will match better the experiments, which cannot be applied in this study due to the long time requirement.

BIBLIOGRAPHY

[1] Arias, S. "An analysis of two-phase flows in conditions relevant to microgravity". (Doctoral dissertation) (Universitat Politècnica de Catalunya. Barcelona. 2011) xi, 5, 6

[2] Arias, S., Montlaur, A. "Numerical study and experimental comparison of two-phase flow generation in a T-junction" *AIAA Journal* . **55** (5), 1565-1574. (2017) 5, 9, 11, 17, 30, 32, 33

[3] Arias, S., Montlaur, A. "Influence of contact angle boundary condition on CFD simulation of T-junction" *Microgravity science and technology*. 1-9. (2018) 9, 17, 18, 34

[4] Bakkali, E. "3D CFD analysis of generation of bubbles in a double T-junction mini channel". (Unpublished bachelor's thesis. Universitat Politècnica de Catalunya. Barcelona. 2018) xi, 5, 6

[5] Donald, F., Bruce, R., Theodore, H. "Viscous flow in pipes". *A brief introduction to fluid mechanics*. (John Wiley & . US. 2004): 313-318.

[6] Hernandez, I. "Comparison between 3D CFD numerical simulations and experimental data in the gas/liquid train generation process in a micro T-junction". (Bachelor's thesis. Universitat Politècnica de Catalunya. Barcelona. 2016) 52-55 63

[7] OpenCFD Ltd (ESI Group). (2004-2018) *OpenFOAM Download*. Available in: https://www.openfoam.com/download/ 15/07/18 41

[8] OpenCFD Ltd (ESI Group). (2016-2018) *OpenFoam: Standard boundary conditions*. Available in: https://www.openfoam.com/documentation/user-guide/standard-boundaryconditions.php#x35-138000A.4 15/07/18 41

[9] OpenCFD Ltd (ESI Group). (2016-2018) *OpenFOAM v5 User Guide: 4.3 Time and data input/output control*. Available in: https://cfd.direct/openfoam/user-guide/controldict/ 15/07/18 48

[10] OpenCFD Ltd (ESI Group). (2016-2018) *OpenFOAM v5 User Guide: 3.4 Running applications in parallel*. Available in: https://cfd.direct/openfoam/user-guide/running-applications-parallel/ 15/07/18 49

[11] OpenCFD Ltd (ESI Group). (2016-2018) *OpenFOAM v5 User Guide: 4.4 Numerical schemes*. Available in: https://cfd.direct/openfoam/user-guide/fvschemes/ 15/07/18 49

[12] OpenCFD Ltd (ESI Group). (2016-2018) *OpenFOAM v5 User Guide: 4.5 Solution and algorithm control*. Available in: https://cfd.direct/openfoam/user-guide/fvsolution/ 15/07/18 49

[13] Paraview (Kitware, Inc.). (2018) *Paraview Download*. Available in: https://www.paraview.org/download/ 15/07/18 59

[14] Rezkallah, K. S. "Weber number based flow-pattern maps for liquid-gas flows at microgravity". *International Journal of Multiphase Flow*. **22** (6), 1265-1270 (1996) 11

[15] Schlichting, H., Gersten, K. "Boundary-Layer Theory". *Springer-Verlag*. 416–419 (2017) 11

[16] Suo, M., Griffith, P. "Two-phase flow in capillary tubes". *Journal of Basic Engineering*. 86, 576-582 (1964) 11

[17] YouTube (comflics). (2012) *OpenFOAM CFD Tutorial*. Available in: https://www.youtube.com/watch?v=AJESwh-QfSo&list=PL63A46D093A83B5CB 15/07/18 41

[18] YouTube (József Nagy). (2015-2018) *OpenFOAM Tutorials*. Available in: https://www.youtube.com/channel/UCjdgpuxuAxH9BqheyE82Vvw/playlists 15/07/18 41

[19] Zuber, N., Findlay, J. "Average Volumetric Concentration in Two-Phase Systems". *Journal of Heat Transfer*, **87**(4), 453–468. (1965) doi:10.1115/1.3689137 33

APPENDICES

APPENDIX A. OPENFOAM

The last version of OpenFOAM can be downloaded for Windows and Mac from [7], with all the steps explaining the installation. Some interesting video tutorials to start learning how to use OpenFOAM can be fount at [17, 18].

A.1. Case file

A description of how to start an OpenFOAM case is explained here, including all the folders. At the beginning of every file there, is a brief description of the version of OpenFOAM, the format and the parameter of the file.

A.1.1. 0 folder

Inside 0 folder we have: ***alpha*** (Figure A.1), ***p_rgh*** (Figure A.2) and ***u*** (Figure A.3). All files inside **0** folder have the same format, first *dimensions* defines the dimensions in a vector as [kg, m, s, K, mol, A, cd], for example m^2s^{-2} would be [0 2 -2 0 0 0 0]. Then the *internalField* defines the initial value of the internal fluid, and *boundaryField* describes the boundary conditions of each surface of the geometry, in this case the geometry is divided in inlet1 (air entrance), inlet2 (water entrance), outlet (exit), walls1 (horizontal pipe), and walls2 (vertical pipe). A list and explanation of the possible boundary conditions are available at [8].

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alpha.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    walls1
    {
        type            fixedValue;
        value           uniform 0;
    }

    walls2
    {
        type            fixedValue;
        value           uniform 0;
    }
    inlet1
    {
        type            fixedValue;
        value           uniform 1;
    }

    inlet2
    {
        type            fixedValue;
        value           uniform 0;
    }

    outlet
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

Figure A.1: Alpha file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet1
    {
        type            zeroGradient;
    }
    inlet2
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    walls1
    {
        type            zeroGradient;
    }
    walls2
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

Figure A.2: Pressure file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet1
    {
        type            fixedValue;
        value           uniform (0.059 0 0);
    }
    inlet2
    {
        type            fixedValue;
        value           uniform (0 -0.318 0);
    }
    outlet
    {
        type            zeroGradient;

    }
    walls1
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    walls2
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}

// ************************************************************************* //
```

Figure A.3: Velocity file

A.1.2. Constant folder

In ***transportProperties***, as we can see in Figure A.4, first in *phases* you write the name of the different phases being, in first place the one that is defined as $\alpha = 1$, in this case the air. Then the *transportModel* (Figure A.4) is defined, in our case the fluids are Newtonian, the dynamic viscosity as *nu* and the density *rho* of each fluid. In addition, the value of the surface tension between the two fluids is defined as *sigma*.

The***turbulenceProperties*** (Figure A.5) just describes the turbulence model in *simulationType*, in this case we are in laminar conditions.

The ***g*** file (Figure A.6) only defines the scalar value of the gravity, since we work under microgravity conditions we can assume the value as 0.
Inside constant folder there is also the *polymesh* folder which contains the mesh description, since this folder is automatically create by OpenFOAM I did not add here the different files.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (air water);

air
{
    transportModel  Newtonian;
    nu              8.163e-06;
    rho             1.225;
}

water
{
    transportModel  Newtonian;
    nu              0.000001;
    rho             1000;
}

sigma               0.072;

// ************************************************************************* //
```

Figure A.4: TransportProperties file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;

// ************************************************************************* //
```

Figure A.5: TurbulenceProperties file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           (0 0 0);  //no gravity considered


// ************************************************************************* //
```

Figure A.6: Gravity file

A.1.3. System folder

In ***controlDict***(Figure A.7) different options related to the time control, data writing and other settings are available.

In *application* you define the solver you want to use to simulate, the *startFrom* controls the start time of the simulation with the different options: *firstTime* earliest time step from the set of time directories, *startTime* time specified by the *startTime* keyword entry, *latestTime* most recent time step from the set of time directories. The *stopAt* controls the end time of the simulation with the options: *endTime* time specified by the *endTime* keyword entry, *writeNow* stops simulation on completion of current time step and writes data, *noWriteNow* stops simulation on completion of current time step and does not write out data, and *nextWrite* stops simulation on completion of next scheduled write time, specified by *writeControl*. *DeltaT* is the time step of the simulation in seconds (all the times are in seconds).

WriteControl controls the timing of writing the output to file with the options: *timeStep* writes data every *writeInterval* time steps, *runTime* writes data every *writeInterval* seconds of simulated time, *adjustableRunTim* writes data every *writeInterval* seconds of simulated time, adjusting the time steps to coincide with the *writeInterval* if necessary (used in cases with automatic time step adjustment), *cpuTime* writes data every *writeInterval* seconds of CPU time, and *clockTime* writes data out every *writeInterval* seconds of real time.

PurgeWrite is the integer representing a limit on the number of time directories that are stored by overwriting time directories on a cyclic basis. For example, if the simulation starts at t=5s and Δt=1s, then with *purgeWrite* 2; data is first written into 2 directories, 6 and 7, then when 8 is written, 6 is deleted, and so on so that only 2 new results directories exists at any time. To disable the purging as in this case by default, *purgeWrite* 0.

WriteFormat specifies the format of the data files as *ascii* by default which means ASCII format, written to *writePrecision* significant figures, and *binary* for binary format.
The *writeCompression* can be *on/off* to specify whether files are compressed with gzip when written. *TimeFormat* and *timePrecision* allow you to choose the format of the naming of the time directories, more information at [9]. The *runTimeModifiable* let you modify the controlDict parameters while running the simulation and it can be *yes/no*. The *adjustTimeStep* will vary the *timeStep* value while the simulation is running in order to have always a Courant number below the *maxCo* defined, and it can be defined as *yes/no*.

In this case, the function *faceSource* was added in order to save the value of alpha every time step, there are two functions since we want to save this value in two sections: at x=7mm and x=8mm as explained in 2.4.. This function lets you save the value of the parameter defined in *fields*. The *outputControl* defines when to save this data, if you put *timeStep* it will save every time step or if you put *outputInterval* you can define this time. In *operation*, the mathematical operation of the value you want to save is defined, in this case we save the average of alpha in the surface defined at *sampledSurfaceDict*.

In ***decomposeParDict*** (Figure A.8) you define the way of decomposing the files if you run the simulations in parallel only. In *numberOfSubdomains* you define the number of

cores and in *method*, the method followed, in this case, the simple, more information about the other cases at [10].

The ***setFields*** (Figure A.9) file is used when you want to define a volume inside your geometry with some characteristic fields. In this case, an air volume is defined at the entrance of the inlet1 in order to help to initialise this problem. At *defaultFieldsValues* you define the value of the field outside of the volume created. In *regions* you define the new volume since the geometry is a pipe, we need a cylinder so the region is *cylinderToCell*. The points p1 and p2 define the center of the initial and final circumference of the cylinder. In addition, the radius is also defined and in *fieldValues* you define the value of the field inside this volume. All the values are in meters.

In ***fvSchemes*** (Figure A.10) dictionary, the numerical schemes for terms, such as derivatives in equations, that are calculated during a simulation is set. The first and second time derivatives are defined in *ddtSchemes*, the gradient in *gradSchemes*, the divergence in *divSchemes*, the Laplacian in *laplacianSchemes*, the cell to face interpolations of values in *interpolationSchemes*, the component of gradient normal to a cell face in *snGradSchemes*, and in *fluxRequired* you can define the fields which require the generation of a flux, more information at [11].

The ***fvSolution*** (Figure A.11)controls the equation solvers, tolerances, and algorithms. In *solvers*, the solver for each field is defined. PIMPLE is the algorithm used in this study, and the *relaxationFactors* controls under-relaxation, a technique used for improving the stability of a computation, which works by limiting the amount which a variable changes from one iteration to the next, more information at [12].

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         2;

deltaT          0.000005;

writeControl    clockTime;

writeInterval   900;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  no;

maxCo           2;
maxAlphaCo      1;

functions
{
    surface8
    {
        type                faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled             true;
        outputControl       timeStep;
        //outputInterval    1800;
        log                 true;
        valueOutput         false;
        source              sampledSurface;
        sourceName          surface8;
        operation           average;
        sampledSurfaceDict
        {
                type        cuttingPlane;
                planeType   pointAndNormal;
                pointAndNormalDict
                {
                        basePoint       (0.008 0.0005 0);
```

```
                        normalVector    (1 0 0);
                }
                source      cells;
                interpolate false;
        }
        fields
        (
            alpha.air
        );
        surfaceFormat dx;
    }

    surface7
    {
        type                faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled             true;
        outputControl       timeStep;
        //outputInterval    1800;
        log                 true;
        valueOutput         false;
        source              sampledSurface;
        sourceName          surface7;
        operation           average;
        sampledSurfaceDict
        {
                type        cuttingPlane;
                planeType   pointAndNormal;
                pointAndNormalDict
                {
                        basePoint       (0.007 0.0005 0);
                        normalVector    (1 0 0);
                }
                source cells;
                interpolate false;
        }
        fields
        (
            alpha.air
        );
        surfaceFormat dx;
    }
}

// ************************************************************************* //
```

Figure A.7: ControlDict file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 4;

method          simple;

simpleCoeffs
{
    n               (2 2 1);
    delta           0.001;
}

hierarchicalCoeffs
{
    n               (1 1 1);
    delta           0.001;
    order           xyz;
}

manualCoeffs
{
    dataFile        "";
}

distributed     no;

roots           ( );


// ************************************************************************* //
```

Figure A.8: Decompose file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue alpha.air 0
);

regions
(
    cylinderToCell
    {
        p1 (0 0.0005 0);
        p2 (0.0005 0.0005 0);
        radius 0.0005;
        fieldValues
        (
            volScalarFieldValue alpha.air 1
        );
    }
);


// ************************************************************************* //
```

Figure A.9: SetFields file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
    default         Gauss linear;
}

divSchemes
{
    div(rhoPhi,U)  Gauss linearUpwind grad(U);
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p_rgh;
    pcorr;
    alpha.air;
}

// ************************************************************************* //
```

Figure A.10: FvSchemes file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.air.*"
    {
        nAlphaCorr      2;
        nAlphaSubCycles 1;
        cAlpha          1;

        MULESCorr       yes;
        nLimiterIter    5;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
       relTol          0;
    }

    "pcorr.*"
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-5;
        relTol          0;
    }

    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          0.05;
    }

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor   no;
    nOuterCorrectors    1;
    nCorrectors         3;
    nNonOrthogonalCorrectors 0;
    cAlpha          1;
    nAlphaCorr      2;
```

```
    nAlphaSubCycles 5;
}

relaxationFactors
{
        "U.*"           1;
        "pcorr.*"       1;
}

// ************************************************************************* //
```

Figure A.11: FvSolutions file

A.1.4. Contact angle

The contact angle boundary condition can be defined in the *alpha* file with the condition *constantContactAngle*. Figure A.12 shows the ***alpha*** file with this condition. *Theta0* corresponds to the value of the contact angle, in openFOAM we should define as 180-value since is defined in the inverse way.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alpha.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    walls1
    {
        type            constantAlphaContactAngle;
        theta0          155;
        limit           gradient;
        value           uniform 0;
    }
```

Figure A.12: Example of *alpha* with the condition of contact angle

A.2. Start a simulation

Once the case file is created you can start the simulation. First open OpenFOAM and enter into the case file. The command to start the simulation is that easy as writing the name of the solver, for example *interFoam*. In this case, since we use *setFieldsDict* and we run in parallel we need to write more commands. The order is the following:

- **setFields** ⇒ to initialise the *setFieldsDict* and create the volume.
- **decomposePar** ⇒ to initialise *decomposeParDict* to decompse the case to run in parallel. This command created various folders (as much as cores you have) named processor1,2,3... and when the simulation is running it will save the results splitted inside.
- **nohup mpirun -np 4 interFoam -parallel>std.txt &** ⇒ the *nohup* command means that you can close the window of the simulation since the simulation will be saved at std.txt. The *mpirun* defines the way to run in parallel, *-np 4* is the number of cores, and *interFoam* the solver.
- **reconstructPar** ⇒ this command is used when the simulation finishes to reconstruct the decomposed folders.

A.3. Post-Processing in Paraview

Paraview can be downloaded from [13].
To see the results in Paraview you need to create inside the case folder a blank file with the extension .foam. In Paraview you need to open this file and if you did not reconstruct the case you should select the option Decomposed case, otherwise Reconstructed case. In Figure A.13 there is a view of the program, you can control the time step you want to see, you can split the geometry to see the behaviour inside, change the field you want to see, save the animation, etc.

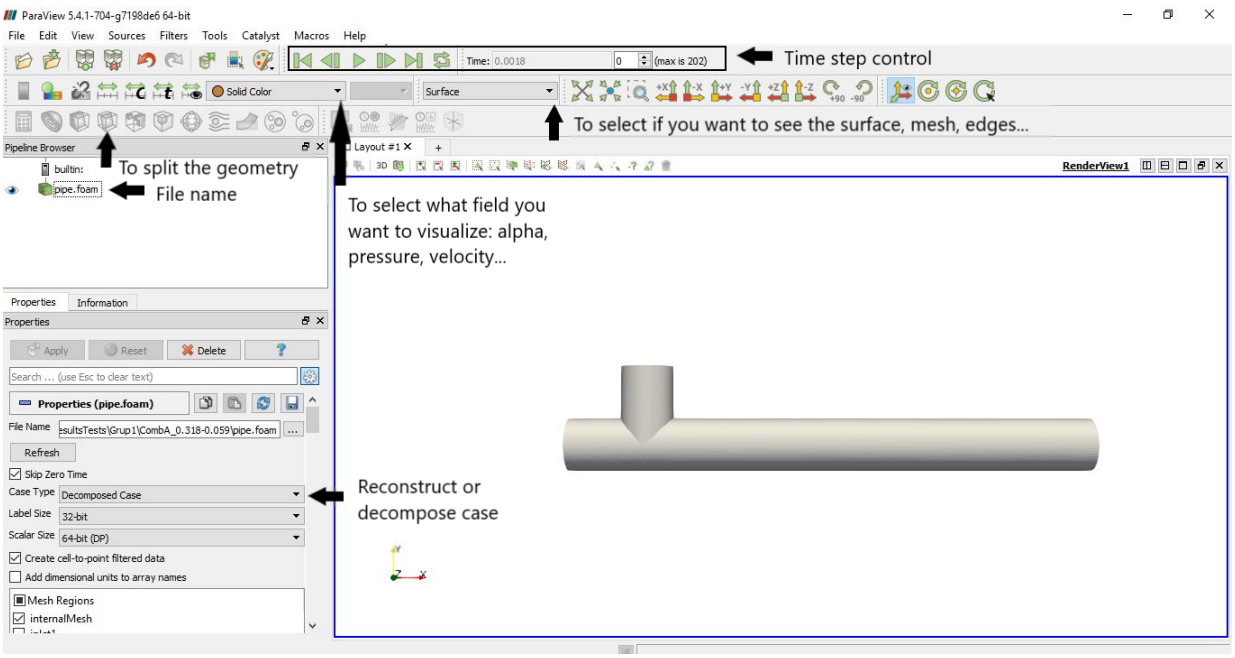

Figure A.13: View and instructions of Paraview

APPENDIX B. RESULTS TABLES

Ref	U_{SL} [m/s]	U_{SG} $[m/s]$	f_{exp} $[Hz]$	f_{simu} $[Hz]$	E_f [%]	σ_f $[Hz]$	$\overline{\sigma_f}$	V_{exp} $[10^{-9}m^3]$	V_{simu} $[10^{-9}m^3]$	E_V [%]	σ_V $[10^{-9}m^3]$	$\overline{\sigma_{V_B}}$
1	0.106	0.276	83.49	71.43	14.4	3.34	0.047	2.596	2.821	-8.7	0.213	0.076
2	0.106	0.363	88.89	81.32	8.5	4.50	0.055	3.206	3.215	-0.2	0.230	0.071
3	0.106	0.516	95.10	94.04	1.1	7.38	0.078	4.260	3.869	9.2	0.343	0.089
4	0.318	0.059	41.81	62.82	-50.3	3.67	0.058	1.118	0.707	36.7	0.043	0.060
5	0.318	0.144	103.72	155.94	-50.3	8.06	0.052	1.093	0.686	37.2	0.047	0.068
6	0.318	0.267	147.78	204.19	-38.1	14.16	0.069	1.422	0.776	45.46	0.078	0.101
7	0.318	0.359	191.93	291.03	-51.6	8.21	0.028	1.483	0.902	39.2	0.080	0.089
8	0.318	0.509	214.43	349.89	-63,2	27,31	0.078	1,867	1,107	40.7	0.120	0.108
9	0.531	0.051	73.28	176.73	-141.2	88,.55	0.501	0.552	0.285	48.3	0.035	0.124
10	0.531	0.136	185.83	360.09	-93.8	33.95	0.094	0.575	0.309	46.2	0.019	0.061
11	0.531	0.259	339.83	621.09	-82.8	14.97	0.024	0.599	0.344	42.6	0.012	0.035
12	0.531	0.352	427.63	393.99	7.87	20.77	0.053	0.648	0.702	-8.2	0.053	0.076
13	0.531	0.505	545.18	424.26	22.2	10.95	0.026	0.729	0.939	-28.7	0.046	0.049

Table B.1: Results and comparison of the bubble frequency and volume between experiments and simulations

Ref	U_{SL} [m/s]	U_{SG} $[m/s]$	U_{Bexp} $[m/s]$	U_{Bsimu} $[m/s]$	E_{U_B} [%]	σ_{U_B} $[m/s]$	$\overline{\sigma_{U_B}}$	L_{Bexp} $[mm]$	L_{Bsimu} $[mm]$	E_{L_B} [%]	σ_{L_B} $[mm]$	$\overline{\sigma_{L_B}}$
1	0.106	0.276	0.415	0.404	2.6	0.009	0.023	4.269	4.635	-8.6	0.340	0.073
2	0.106	0.363	0.521	0.500	4.1	0.005	0.011	5.035	5.350	-6.2	0.302	0.056
3	0.106	0.516	0.713	0.685	3.9	0.021	0.031	6.740	6.710	0.5	0.526	0.078
4	0.318	0.059	0.350	0.396	-12.9	0.015	0.037	2.120	1.514	28.6	0.079	0.052
5	0.318	0.144	0.514	0.491	4.4	0.005	0.009	2.080	1.508	27.5	0.075	0.049
6	0.318	0.267	0.684	0.578	15.53	0.004	0.006	2.538	1.680	33.8	0.126	0.074
7	0.318	0.359	0.807	0.757	6.2	0.007	0.009	3.000	1.946	35.1	0.132	0.068
8	0.318	0.509	1.000	0.970	2.9	0.015	0.015	3.444	2.388	30.7	0.211	0.088
9	0.531	0.051	0.602	0.727	-20.8	0.020	0.028	0.961	0.982	-2.1	0.034	0.035
10	0.531	0.136	0.730	0.843	-15.5	0.014	0.017	1.240	1.025	17.4	0.018	0.017
11	0.531	0.259	0.869	0.977	-12.3	0.014	0.014	1.370	1.070	21.9	0.019	0.018
12	0.531	0.352	0.960	1.046	-8.9	0.026	0.025	1.321	1.788	-35.3	0.118	0.066
13	0.531	0.505	1.241	1.297	-4.5	0.043	0.033	1.592	2.342	-47.1	0.093	0.039

Table B.2: Results and comparison of the bubble velocity and longitude between experiments and simulations

APPENDIX C. MATLAB CODES

C.1. Validations scripts

The Matlab script used to calculate the frequency, length, and velocity of the bubbles from the alpha files saved by OpenFOAM, for the Validations section 3, is presented here. This code was developed by Isaac Hernandez in [6], we added the calculation of the standard deviation.

```
close all;
clear all;

%% Analysis of the data in surface x=8mm
load surf_mon_8.dat
ref=surf_mon_8;

area=pi*((1e-3)/2)^2;    % Area of 1mm circumference
airref = ref(:,2);       % Air volume fraction
wref = 1-airref;         % Water volume fraction
tref=ref(:,1);           %Time

% Find the initial and final points of every bubble:
i=1;
k=1;
init=true;
valor=0.0015;  %minimum value of alpha considering a bubble
while(i<=length(airref))
    done=false;
    if(airref(i)>=valor&&init==true)
        points(k)=i;
        k=k+1;
        init=false;
        done=true;
    end
    if(airref(i)<=valor&&init==false&&done==false)
        points(k)=i;
        k=k+1;
        init=true;
    end
    i=i+1;
end

% Remove odd numbers:
if(rem(length(points),2)==1)
    points(k-1)=[];
end
```

```
38
39 % Separation between initial and final points:
40 i=1;
41 k=1;
42 while(i<=length(points))
43     ipoints(k)=points(i);
44     i=i+1;
45     fpoints(k)=points(i);
46     i=i+1;
47     k=k+1;
48 end
49
50 % Bubble area:
51 i=1;
52 while(i<=length(ipoints))
53     a(i)=trapz(tref(ipoints(i):fpoints(i)),airref(ipoints(i)
           :fpoints(i)));
54     av(i)=a(i)*area;
55     i=i+1;
56 end
57
58 % Remove little bubbles (simulation errors):
59 i=1;
60 while(i<=length(av))
61     if(av(i)<=10e-11)
62         fpoints(i)=[];
63         ipoints(i)=[];
64         av(i)=[];
65         a(i)=[];
66         i=i-1;
67     end
68     i=i+1;
69 end
70
71 % Bubble frequency:
72 i=1;
73 while(i<=length(ipoints)-1)
74     periode(i)=tref(ipoints(i+1))-tref(ipoints(i));
75     fi(i)=1/periode(i);
76     i=i+1;
77 end
78
79 %% Analysis of the data in surface x=7mm
80
81 load surf_mon_7.dat
82 ref7=surf_mon_7;
83 airref7=ref7(:,2);  %Air volume fraction
84 wref7 = 1-airref7;  %Water volume fraction
```

```
tref7=ref7(:,1);      %Time

% Find the initial and final points of every bubble:
i=1;
k=1;
init=true;
valor=0.0015;
while(i<=length(airref7))
    done=false;
    if(airref7(i)>=valor&&init==true)
        points7(k)=i;
        k=k+1;
        init=false;
        done=true;
    end
    if(airref7(i)<=valor&&init==false&&done==false)
        points7(k)=i;
        k=k+1;
        init=true;
    end
    i=i+1;
end

% Remove odd numbers:
if(rem(length(points7),2)==1)
    points7(k-1)=[];
end

% Separation between initial and final points:
i=1;
k=1;
while(i<=length(points7))
    ipoints7(k)=points7(i); %Initial point
    i=i+1;
    fpoints7(k)=points7(i); %Final point
    i=i+1;
    k=k+1;
end

% Bubble area:
i=1;
while(i<=length(ipoints7))
    a7(i)=trapz(tref7(ipoints7(i):fpoints7(i)),airref7(
        ipoints7(i):fpoints7(i)));
    av7(i)=a7(i)*area;
    i=i+1;
end

```

```
% Remove little bubbles (simulation errors):
i=1;
while(i<=length(av7))
    if(av7(i)<=10e-11)
        fpoints7(i)=[];
        ipoints7(i)=[];
        av7(i)=[];
        a7(i)=[];
        i=i-1;
    end
    i=i+1;
end

%% Parameter calculations

% Bubbles velocity:
i=1;
while(i<=length(ipoints7)
    tbetween(i)=tref(ipoints(i))-tref7(ipoints7(i));
    if(tbetween(i)<0)
        tbetween(i)=tref(ipoints(i+1))-tref7(ipoints7(i));
    end
    Ug(i)=1e-3/tbetween(i);
    if(Ug(i)<0)
        Ug(i)=[];
    end
    i=i+1;

end

% Bubbles length:
i=1;
while(i<=length(Ug))
    tbubbles(i)=tref(fpoints(i))-tref(ipoints(i));
    L(i)=Ug(i)*tbubbles(i);
    i=i+1;
end

% Vector taking the last X bombolles
% Define the number of bubbles you want to calculate the
    parameters
num_total=5;
% Total number of bubbles you will use to calculate the
    standar deviation
num_bombolles=2;
i=1;
while(i<=num_total)
    fi2(i)=fi(i);
```

```
178       U2(i)=Ug(i);
179       L2(i)=L(i);
180       i=i+1;
181   end
182
183   % Vectors with the last X bubbles
184   k=1;
185   for  i=1:length(fi2)
186       if(i>(num_total-num_bombolles))
187           finova(k)=fi2(i);
188           Ugnova(k)=U2(i);
189           Lnova(k)=L2(i);
190           k=k+1;
191       end
192       i=i+1;
193   end
194
195   % Mean calculation
196   frequency2=mean(finova)
197   longitude2=mean(Lnova)*1000
198   velocity2=mean(Ugnova)
199
200   % Standard deviation calculation
201   for  i=1:length(finova)
202       f(i)=(finova(i)-frequency2)^2;
203       l(i)=(Lnova(i)*1000-longitude2)^2;
204       u(i)=(Ugnova(i)-velocity2)^2;
205   end
206
207   df=sqrt(sum(f)/(num_bombolles-1))
208   dl=sqrt(sum(l)/(num_bombolles-1))
209   du=sqrt(sum(u)/(num_bombolles-1))
210
211   % To save the data, and use later for the plot script
212   % If you are analysing Mesh1
213   fiMesh1=fi2;
214   UMesh1=U2;
215   LMesh1=L2;
216   save('fiMesh1.mat','fiMesh1');
217   save('LMesh1.mat','LMesh1');
218   save('UMesh1.mat','UMesh1');
219
220   % If you are analysing Mesh2
221   fiMesh2=fi2;
222   UMesh2=U2;
223   LMesh2=L2;
224   save('fiMesh2.mat','fiMesh2');
225   save('LMesh2.mat','LMesh2');
```

```
save('UMesh2.mat','UMesh2');

% If you are analysing Mesh3
fiMesh3=fi2;
UMesh3=U2;
LMesh3=L2;
save('fiMesh3.mat','fiMesh3');
save('LMesh3.mat','LMesh3');
save('UMesh3.mat','UMesh3');

%% Graphics

% Plot with all the bubbles:
figure(1)
subplot(2,2,1)
plot(tref,airref,'g-')
xlabel('time [s]')
ylabel('fraction of air')
title('x=0.008m')
subplot(2,2,2)
plot(fi)
xlabel('bubble')
ylabel('frequency [Hz]')
subplot(2,2,3)
plot(L*1000)
xlabel('bubble')
ylabel('Longitude [mm]')
subplot(2,2,4)
plot(Ug,'-r')
xlabel('bubble')
ylabel('velocity [m/s]')

% Plot with the number of bubbles you cutted
figure(2)
subplot(2,2,1)
plot(tref,airref,'g-')
xlabel('time [s]')
ylabel('fraction of air')
title('x=0.008m')
subplot(2,2,2)
plot(fi2)
xlabel('bubble')
ylabel('frequency [Hz]')
subplot(2,2,3)
plot(L2)
xlabel('bubble')
ylabel('longitude [m]')
subplot(2,2,4)
```

```
274 plot(U2,'-r')
275 xlabel('bubble')
276 ylabel('velocity [m/s]')
```

The script to generate the comparative graphics of the different meshes is the following:

```
1  close all;
2  clear all;
3
4  % Load the files with the data
5  load('fiMesh1.mat')
6  load('fiMesh2.mat')
7  load('fiMesh3.mat')
8  load('LMesh1.mat')
9  load('LMesh2.mat')
10 load('LMesh3.mat')
11 load('UMesh1.mat')
12 load('UMesh2.mat')
13 load('UMesh3.mat')
14
15 % We take the last 6 bubbles
16 k=1;
17 for i=1:length(LMesh1)
18     if(i>=11)
19         fiMesh11(k)=fiMesh1(i);
20         fiMesh22(k)=fiMesh2(i);
21         fiMesh33(k)=fiMesh3(i);
22         UMesh11(k)=UMesh1(i);
23         UMesh22(k)=UMesh2(i);
24         UMesh33(k)=UMesh3(i);
25         LMesh11(k)=LMesh1(i);
26         LMesh22(k)=LMesh2(i);
27         LMesh33(k)=LMesh3(i);
28         k=k+1;
29     end
30 end
31
32 % Means of the last 6 bubbles
33 F1=mean(fiMesh11);
34 F2=mean(fiMesh22);
35 F3=mean(fiMesh33);
36 L1=mean(LMesh11*1000);
37 L2=mean(LMesh22*1000);
38 L3=mean(LMesh33*1000);
39 U1=mean(UMesh11);
40 U2=mean(UMesh22);
41 U3=mean(UMesh33);
42
```

```
% Errors calculation
ErrorF1= (F1-F3)/F3*100
ErrorV1=(U1-U3)/U3*100
ErrorL1=(L1-L3)/L3*100
ErrorF2= (F2-F3)/F3*100
ErrorV2=(U2-U3)/U3*100
ErrorL2=(L2-L3)/L3*100

% Plots of the graphics
figure(1)
subplot(3,1,1)
plot(fiMesh1,':k')
hold on;
plot(fiMesh2,'--k')
hold on;
plot(fiMesh3,'-k')
xlabel('Bubble')
ylabel('Frequency [Hz]')
subplot(3,1,2)
plot(LMesh1*1000,':k')
hold on;
plot(LMesh2*1000,'--k')
hold on;
plot(LMesh3*1000,'-k')
xlabel('Bubble')
ylabel('Longitude [mm]')
leg=legend('200 000','400 000','600 000');
title(leg,'Number of cells')
subplot(3,1,3)
plot(UMesh1,':k')
hold on;
plot(UMesh2,'--k')
hold on;
plot(UMesh3,'-k')
xlabel('Bubble')
ylabel('Velocity [m/s]')
```

C.2. Results scripts

The Matlab script used to calculate the frequency, length, velocity, and volume of the bubble from the alpha files saved by OpenFOAM, for the Results section 4, is presented here.

```
close all;
clear all

```

```
%% Analysis of the data in surface x=8mm

load surf_mon_8.dat
ref=surf_mon_8;
area=pi*((1e-3)/2)^2;    % Area of 1mm circumference
airref = ref(:,2);       % Air volume fraction
wref = 1-airref;         % Water volume fraction
tref=ref(:,1);           %Time

% Find the initial and final points of every bubble:
i=1;
k=1;
init=true;
valor=0.0015; % minimum value of alpha considering a bubble
while(i<=length(airref))
    done=false;
    if(airref(i)>=valor&&init==true)
        points(k)=i;
        k=k+1;
        init=false;
        done=true;
    end
    if(airref(i)<=valor&&init==false&&done==false)
        points(k)=i;
        k=k+1;
        init=true;
    end
    i=i+1;
end

% Remove odd numbers:
if(rem(length(points),2)==1)
    points(k-1)=[];
end

% Separation between initial and final points:
i=1;
k=1;
while(i<=length(points))
    ipoints(k)=points(i);
    i=i+1;
    fpoints(k)=points(i);
    i=i+1;
    k=k+1;
end

% Bubble area:
i=1;
```

```
52 while(i<=length(ipoints))
53     a(i)=trapz(tref(ipoints(i):fpoints(i)),airref(ipoints(i)
       :fpoints(i)));
54     av(i)=a(i)*area;
55     i=i+1;
56 end
57
58 % Remove little bubbles (simulation errors):
59 i=1;
60 while(i<=length(av))
61     if(av(i)<=10e-11)
62         fpoints(i)=[];
63         ipoints(i)=[];
64         av(i)=[];
65         a(i)=[];
66         i=i-1;
67     end
68     i=i+1;
69 end
70
71 % Bubble frequency:
72 i=1;
73 while(i<=length(ipoints)-1)
74     periode(i)=tref(ipoints(i+1))-tref(ipoints(i));
75     fi(i)=1/periode(i);
76     i=i+1;
77 end
78
79 %% Analysis of the data in surface x=7mm
80
81 load surf_mon_7.dat
82 ref7=surf_mon_7;
83 airref7=ref7(:,2);  % Air volume fraction
84 wref7 = 1-airref7;  % Water volume fraction
85 tref7=ref7(:,1);    % Time
86
87 % Find the initial and final points of every bubble:
88 i=1;
89 k=1;
90 init=true;
91 valor=0.0015;
92 while(i<=length(airref7))
93     done=false;
94     if(airref7(i)>=valor&&init==true)
95         points7(k)=i;
96         k=k+1;
97         init=false;
98         done=true;
```

```
99      end
100     if(airref7(i)<=valor&&init==false&&done==false)
101         points7(k)=i;
102         k=k+1;
103         init=true;
104     end
105     i=i+1;
106 end
107
108 % Remove odd numbers:
109 if(rem(length(points7),2)==1)
110     points7(k-1)=[];
111 end
112
113 % Separation between initial and final points:
114 i=1;
115 k=1;
116 while(i<=length(points7))
117     ipoints7(k)=points7(i); % Initial point
118     i=i+1;
119     fpoints7(k)=points7(i); % Final point
120     i=i+1;
121     k=k+1;
122 end
123
124 % Bubble area:
125 i=1;
126 while(i<=length(ipoints7))
127     a7(i)=trapz(tref7(ipoints7(i):fpoints7(i)),airref7(
          ipoints7(i):fpoints7(i)));
128     av7(i)=a7(i)*area;
129     i=i+1;
130 end
131
132 % Remove little bubbles (simulation errors):
133 i=1;
134 while(i<=length(av7))
135     if(av7(i)<=10e-11)
136         fpoints7(i)=[];
137         ipoints7(i)=[];
138         av7(i)=[];
139         a7(i)=[];
140         i=i-1;
141     end
142     i=i+1;
143 end
144
145 %% Parameter calculations
```

```
146
147 % Bubble velocity and volume:
148 i=1;
149 while(i<=length(ipoints))
150     tbetween(i)=tref(ipoints(i))-tref7(ipoints7(i));
151     Ug(i)=1e-3/tbetween(i); % Velocity
152     volume(i)=av(i)*Ug(i);  % Volume
153     i=i+1;
154 end
155
156 % Bubble length:
157 i=1;
158 while(i<=length(ipoints))
159     tbubbles(i)=tref(fpoints(i))-tref(ipoints(i));
160     L(i)=Ug(i)*tbubbles(i);
161     vol(i)=av(i)*L(i);
162     i=i+1;
163 end
164
165 % Vector with the total bubbles that we will use to
    calculate
166 % the standard deviation:
167
168 num_total=6;       % Number of total bubbles
169 num_bombolles=6;  % Number of bubbles used to calculate the
    parameters
170 i=1;
171 while(i<=num_total)
172     fi2(i)=fi(i);
173     U2(i)=Ug(i);
174     L2(i)=L(i);
175     V2(i)=volume(i);
176     i=i+1;
177 end
178
179 % Vector with the last 6 bubbles:
180 k=1;
181 for i=1:length(fi2)
182     if(i>(num_total-num_bombolles))
183         finova(k)=fi2(i);
184         Ugnova(k)=U2(i);
185         Lnova(k)=L2(i);
186         Vnova(k)=V2(i);
187         k=k+1;
188     end
189     i=i+1;
190 end
191
```

```
% Frequency, longitude, velocity and volume final values:
frequency2=mean(finova)
longitude2=mean(Lnova)*1000
velocity2=mean(Ugnova)
volume2=mean(Vnova)

% Standard deviation calculation:
for i=1:length(finova)
    f(i)=(finova(i)-frequency2)^2;
    l(i)=(Lnova(i)*1000-longitude2)^2;
    u(i)=(Ugnova(i)-velocity2)^2;
    v(i)=(Vnova(i)-volume2)^2;
end
df=sqrt(sum(f)/(num_bombolles-1))
dl=sqrt(sum(l)/(num_bombolles-1))
du=sqrt(sum(u)/(num_bombolles-1))
dv=sqrt(sum(v)/(num_bombolles-1))

% To save the parameters in a file:
comb12=[frequency2,longitude2,velocity2,volume2];
dev12=[df,dl,du,dv];
save('comb12.mat','comb12');
save('dev12.mat','dev12');

%% Plots
figure(1)
subplot(2,2,1) % Alpha plot
plot(tref,airref,'g-')
xlabel('time [s]')
ylabel('fraction of air')
title('x=0.008m')
subplot(2,2,2) % Frequency plot
plot(fi)
xlabel('bubble')
ylabel('frequency [Hz]')
subplot(2,2,3)  % Longitude plot
plot(L*1000)
xlabel('bubble')
ylabel('Longitude [mm]')
subplot(2,2,4)  % Velocity plot
plot(Ug,'-r')
xlabel('bubble')
ylabel('velocity [m/s]')
```

The script used to generate the graphics with the experimental and simulated data comparison is the following:

```
clear all;
```

```
close all;

% Load all files

% GROUP1
load('comb1.mat')
load('comb2.mat')
load('comb3.mat')
load('comb4.mat')
load('comb5.mat')
load('dev1.mat')
load('dev2.mat')
load('dev3.mat')
load('dev4.mat')
load('dev5.mat')

% GROUP2
load('comb6.mat')
load('comb7.mat')
load('comb8.mat')
load('comb9.mat')
load('comb10.mat')
load('dev6.mat')
load('dev7.mat')
load('dev8.mat')
load('dev9.mat')
load('dev10.mat')

% GROUP3
load('comb13.mat')
load('comb14.mat')
load('comb15.mat')
load('dev13.mat')
load('dev14.mat')
load('dev15.mat')

%% Parametres

% Superficial gas and liquid velocity for each group

% Group1
Usg1=[0,0.059,0.144, 0.267, 0.359,0.509];
Usl1=0.318*ones(1,6); Usl(1)=0;

% Group2
Usg2=[0,0.059,0.144, 0.267, 0.359,0.509];
Usl2=0.531*ones(1,6); Usl(1)=0;

```

```
% Group3
Usg3=[0,0.276,0.363,0.516];
Usl3=0.106*ones(1,4);  Usl(1)=0;
diametre=0.001;

% Experimental data for each group
fE1=[0,41.806,103.721,147.778,191.930,214.425]; % Frequency
lE1=[0,2.120,2.080,2.538,3.000,3.444]/1000;     % Longitude
uE1=[0,0.350,0.514,0.684,0.807,1.000];          % Velocity
vE1=[0,1.118,1.093,1.422,1.483,1.867]*10^-9;    % Volume

fE2=[0,73.282,185.832,339.827,427.632,545.167];
lE2=[0,0.961,1.240,1.370,1.321,1.592]/1000;
uE2=[0,0.602,0.730,0.869,0.960,1.241];
vE2=[0,0.552,0.575,0.599,0.648,0.729]*10^-9;

fE3=[0,83.486,88.889,95.103];
lE3=[0,4.269,5.035,6.740]/1000;
uE3=[0,0.415,0.521,0.713];
vE3=[0,2.596,3.206,4.260]*10^-9;

% Simulaton data and standard deviations for each group
f1=[0,comb1(1),comb2(1),comb3(1),comb4(1),comb5(1)];
l1=[0,comb1(2),comb2(2),comb3(2),comb4(2),comb5(2)]/1000;
u1=[0,comb1(3),comb2(3),comb3(3),comb4(3),comb5(3)];
v1=[0,comb1(4),comb2(4),0.78542*10^-9,comb4(4),comb5(4)];
devf1=[dev1(1),dev2(1),dev3(1),dev4(1),dev5(1)]/2;
devl1=[dev1(2),dev2(2),dev3(2),dev4(2),dev5(2)]/(2*1000);
devu1=[dev1(3),dev2(3),dev3(3),dev4(3),dev5(3)]/2;
devv1=[dev1(4),dev2(4),dev3(4),dev4(4),dev5(4)]/2;

f2=[0,comb6(1),comb7(1),comb8(1),comb9(1),comb10(1)];
l2=[0,comb6(2),comb7(2),comb8(2),comb9(2),comb10(2)]/1000;
u2=[0,comb6(3),comb7(3),comb8(3),comb9(3),comb10(3)];
v2=[0,comb6(4),comb7(4),comb8(4),comb9(4),comb10(4)];
devf2=[dev6(1),dev7(1),dev8(1),dev9(1),dev10(1)]/2;
devl2=[dev6(2),dev7(2),dev8(2),dev9(2),dev10(2)]/(2*1000);
devu2=[dev6(3),dev7(3),dev8(3),dev9(3),dev10(3)]/2;
devv2=[dev6(4),dev7(4),dev8(4),dev9(4),dev10(4)]/2;

f3=[0,comb13(1),comb14(1),comb15(1)];
l3=[0,comb13(2),comb14(2),comb15(2)]/1000;
u3=[0,comb13(3),comb14(3),comb15(3)];
v3=[0,comb13(4),comb14(4),comb15(4)];
devf3=[dev13(1),dev14(1),dev15(1)]/2;
devl3=[dev13(2),dev14(2),dev15(2)]/(2*1000);
devu3=[dev13(3),dev14(3),dev15(3)]/2;
devv3=[dev13(4),dev14(4),dev15(4)]/2;
```

```
98
99  %% Frequency plot
100
101 % Characterisation of the frequency equation for
        experimental
102 % and simulated data of each group
103 exponential1 = fittype('fsat*(1-exp(-ao*Usg1/fsat))','
        coefficients',{'fsat','ao'},'dependent',{'f1'},'
        independent',{'Usg1'});
104 exponentialE1 = fittype('fsat*(1-exp(-ao*Usg1/fsat))','
        coefficients',{'fsat','ao'},'dependent',{'fE1'},'
        independent',{'Usg1'});
105 f_eq1=fit(Usg1',f1',exponential1)
106 fE_eq1=fit(Usg1',fE1',exponentialE1)
107
108 exponential2 = fittype('fsat*(1-exp(-ao*Usg2/fsat))','
        coefficients',{'fsat','ao'},'dependent',{'f2'},'
        independent',{'Usg2'});
109 exponentialE2 = fittype('fsat*(1-exp(-ao*Usg2/fsat))','
        coefficients',{'fsat','ao'},'dependent',{'fE2'},'
        independent',{'Usg2'});
110 f_eq2=fit(Usg2',f2',exponential2)
111 fE_eq2=fit(Usg2',fE2',exponentialE2)
112
113 exponential3 = fittype('fsat*(1-exp(-ao*Usg3/fsat))','
        coefficients',{'fsat','ao'},'dependent',{'f3'},'
        independent',{'Usg3'});
114 exponentialE3= fittype('fsat*(1-exp(-ao*Usg3/fsat))','
        coefficients',{'fsat','ao'},'dependent',{'fE3'},'
        independent',{'Usg3'});
115 f_eq3=fit(Usg3',f3',exponential3)
116 fE_eq3=fit(Usg3',fE3',exponentialE3)
117
118 figure (1)
119 xlim([0,0.6])
120 ylim([0,700])
121 hold on;
122 plot(f_eq1,'--k');
123 hold on;
124 plot(fE_eq1,'-k')
125 hold on;
126 plot(f_eq2,'--k')
127 hold on;
128 plot(fE_eq2,'-k')
129 hold on;
130 plot(f_eq3,'--k')
131 hold on;
132 plot(fE_eq3,'-k')
```

```
hold on;
h(1)=errorbar(Usg1(2:6),f1(2:6),devf1,'o','MarkerEdgeColor','k','Color','k');
hold on;
h(2)=plot(Usg1(2:6),fE1(2:6),'o','MarkerEdgeColor','k','MarkerFaceColor','k');
hold on;
h(3)=errorbar(Usg2(2:6),f2(2:6),devf2,'s','MarkerEdgeColor','k','Color','k');
hold on;
h(4)=plot(Usg2(2:6),fE2(2:6),'s','MarkerEdgeColor','k','MarkerFaceColor','k');
hold on;
h(5)=errorbar(Usg3(2:4),f3(2:4),devf3,'d','MarkerEdgeColor','k','Color','k');
hold on;
h(6)=plot(Usg3(2:4),fE3(2:4),'d','MarkerEdgeColor','k','MarkerFaceColor','k');
hold on;
xlim([0,0.6])
ylim([0,700])
xlabel('U_{SG} [m/s]')
ylabel('f [1/s]')
legend(h,'Sim 0.318 m/s','Exp','Sim 0.531 m/s','Exp','Sim 0.106 m/s','Exp','Location','northwest')
legend('boxoff')

%% Volume plot
for i=2:length(Usg1)
    Vb1(i)=Usg1(i)/(f1(i)*diametre);
    Vb2(i)=Usg2(i)/(f2(i)*diametre);
    VbE1(i)=Usg1(i)/(fE1(i)*diametre);
    VbE2(i)=Usg2(i)/(fE2(i)*diametre);
end
for i=2:length(Usg3)
    Vb3(i)=Usg3(i)/(f3(i)*diametre);
    VbE3(i)=Usg3(i)/(fE3(i)*diametre);
end

% Dimensionless volume and standard deviation
v_adim1=v1./(pi/4*(diametre)^3);
devv_adim1=devv1./(pi/4*(diametre)^3);
v_adimE1=vE1./(pi/4*(diametre)^3);
funct2=@(x) x;

v_adim2=v2./(pi/4*(diametre)^3);
devv_adim2=devv2./(pi/4*(diametre)^3);
v_adimE2=vE2./(pi/4*(diametre)^3);
```

```
174
175 v_adim3=v3./(pi/4*(diametre)^3);
176 devv_adim3=devv3./(pi/4*(diametre)^3);
177 v_adimE3=vE3./(pi/4*(diametre)^3);
178
179 figure (2)
180 h(1)=errorbar(Vb1(2:6),v_adim1(2:6),devv_adim1,'o','
    MarkerEdgeColor','k','Color','k');
181 hold on;
182 h(2)=plot(VbE1(2:6),v_adimE1(2:6),'o','MarkerEdgeColor','k',
    'MarkerFaceColor','k');
183 hold on;
184 fplot(funct2,[0,6],'k')
185 hold on;
186 h(3)=errorbar(Vb2(2:6),v_adim2(2:6),devv_adim2,'s','
    MarkerEdgeColor','k','Color','k');
187 hold on;
188 h(4)=plot(VbE2(2:6),v_adimE2(2:6),'s','MarkerEdgeColor','k',
    'MarkerFaceColor','k');
189 hold on;
190 h(5)=errorbar(Vb3(2:4),v_adim3(2:4),devv_adim3,'d','
    MarkerEdgeColor','k','Color','k');
191 hold on;
192 h(6)=plot(VbE3(2:4),v_adimE3(2:4),'d','MarkerEdgeColor','k',
    'MarkerFaceColor','k');
193 hold on;
194 xlim([0 6])
195 xlabel('U_{SG}/f*\phi_{C}')
196 ylabel('$\bar{V}_{B}$','Interpreter','Latex','fontsize',14)
197 legend(h,'Sim 0.318 m/s','Exp','Sim 0.531 m/s','Exp','Sim
    0.106 m/s','Exp','Location','southeast')
198 legend('boxoff')
199
200 %% Velocity plot
201
202 % Mixture superficial velocity
203 Um1=Usl1+Usg1; Um1(1)=0;
204 Um2=Usl2+Usg2; Um2(1)=0;
205 Um3=Usl3+Usg3; Um3(1)=0;
206 Um=[Um1,Um2(2:6),Um3(2:4)];
207
208 % Characterisation of the velocity equation for experimental
209 % and simulated data
210 uSimu=[u1,u2(2:6),u3(2:4)];
211 uExp=[uE1,uE2(2:6),uE3(2:4)];
212 velocity_eq = fittype('Co*Um','coefficients',{'Co'},'
    dependent',{'uSimu'},'independent',{'Um'});
213 velocity_eqE = fittype('Co*Um','coefficients',{'Co'},'
```

```
    dependent',{'uExp'},'independent',{'Um'});
v_eq1=fit(Um',uSimu',velocity_eq);
vE_eq1=fit(Um',uExp',velocity_eqE);

figure(3)
axis([0 1 0 1.2])
plot(vE_eq1,'-k')
hold on;
plot(v_eq1,'--k')
hold on;
h(1)=errorbar(Um1(2:6),u1(2:6),devu1,'o','MarkerEdgeColor','
    k','Color','k');
hold on;
h(2)=plot(Um1(2:6),uE1(2:6),'o','MarkerEdgeColor','k','
    MarkerFaceColor','k');
hold on;
h(3)=errorbar(Um2(2:6),u2(2:6),devu2,'s','MarkerEdgeColor','
    k','Color','k');
hold on;
h(4)=plot(Um2(2:6),uE2(2:6),'s','MarkerEdgeColor','k','
    MarkerFaceColor','k');
hold on;
h(5)=errorbar(Um3(2:4),u3(2:4),devu3,'d','MarkerEdgeColor','
    k','Color','k');
hold on;
h(6)=plot(Um3(2:4),uE3(2:4),'d','MarkerEdgeColor','k','
    MarkerFaceColor','k');
hold on;
axis([0 1 0 1.2])
xlabel('U_{M} [m/s]')
ylabel('U_{B} [m/s]')
legend(h,'Sim 0.318 m/s','Exp','Sim 0.531 m/s','Exp','Sim
    0.106 m/s','Exp','Location','southeast')
legend('boxoff')

%% Longitude plot

% Dimensionless bubble length and standard deviation
l_adim1=l1/diametre;
lE_adim1=lE1/diametre;
devl_adim1=devl1/diametre;

l_adim2=l2/diametre;
lE_adim2=lE2/diametre;
devl_adim2=devl2/diametre;

l_adim3=l3/diametre;
lE_adim3=lE3/diametre;
```

```
254 devl_adim3=devl3/diametre;
255
256 % Characterisation of the longitude equation for
      experimental
257 % and simulated data
258 Vb=[Vb1,Vb2(2:6),Vb3(2:4)];
259 VbE=[VbE1,VbE2(2:6),VbE3(2:4)];
260 l_adim=[l_adim1,l_adim2(2:6),l_adim3(2:4)];
261 lE_adim=[lE_adim1,lE_adim2(2:6),lE_adim3(2:4)];
262 long_eq = fittype('C1+C2*Vb','coefficients',{'C1','C2'},'
      dependent',{'l_adim'},'independent',{'Vb'});
263 long_eqE = fittype('C1+C2*VbE','coefficients',{'C1','C2'},'
      dependent',{'lE_adim'},'independent',{'VbE'});
264 l_eq=fit(Vb','l_adim',long_eq);
265 lE_eq=fit(VbE','lE_adim',long_eqE);
266
267 figure(4)
268 xlim([0 6])
269 ylim([0 8])
270 plot(l_eq,'--k')
271 hold on;
272 plot(lE_eq,'-k')
273 hold on;
274 h(1)=errorbar(Vb1(2:6),l_adim1(2:6),devl_adim1,'o','
      MarkerEdgeColor','k','Color','k');
275 hold on;
276 h(2)=plot(VbE1(2:6),lE_adim1(2:6),'o','MarkerEdgeColor','k',
      'MarkerFaceColor','k');
277 hold on;
278 h(3)=errorbar(Vb2(2:6),l_adim2(2:6),devl_adim2,'s','
      MarkerEdgeColor','k','Color','k');
279 hold on;
280 h(4)=plot(VbE2(2:6),lE_adim2(2:6),'s','MarkerEdgeColor','k',
      'MarkerFaceColor','k');
281 hold on;
282 h(5)=errorbar(Vb3(2:4),l_adim3(2:4),devl_adim3,'d','
      MarkerEdgeColor','k','Color','k');
283 hold on;
284 h(6)=plot(VbE3(2:4),lE_adim3(2:4),'d','MarkerEdgeColor','k',
      'MarkerFaceColor','k');
285 hold on;
286 xlim([0 6])
287 ylim([0 8])
288 xlabel('U_{SG}/ f *\phi_{C}')
289 ylabel('$\bar{L}_{B}$','Interpreter','Latex','fontsize',14)
290 legend(h,'Sim 0.318 m/s','Exp','Sim 0.531 m/s','Exp','Sim
      0.106 m/s','Exp','Location','southeast')
291 legend('boxoff')
```